TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THÔNG TIN
HỮU NGHỊ VIỆT HÀN

KHOA KHOA HỌC MÁY TÍNH

ĐỒ ÁN TỐT NGHIỆP
HỆ THỐNG THÔNG TIN

Tên đề tài: Nghiên cứu mô hình MVC trong lập trình .NET để xây dựng website đăng ký mua giáo trình qua mạng

Sinh viên thực hiện: Nguyễn Trọng Trí

Lớp: HT02

Niên khóa: 2008 - 2011

Giảng viên hướng dẫn: Ths.Nguyễn Quang Vũ

Đà nẵng, tháng 7 năm 2011

TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THÔNG TIN
HỮU NGHỊ VIỆT HÀN

KHOA KHOA HỌC MÁY TÍNH

ĐỒ ÁN TỐT NGHIỆP
HỆ THỐNG THÔNG TIN

Tên đề tài: Nghiên cứu mô hình MVC trong lập trình .NET để xây dựng website đăng ký mua giáo trình qua mạng

Sinh viên thực hiện: Nguyễn Trọng Trí
Lớp: HT02
Niên khóa: 2008 - 2011
Giảng viên hướng dẫn: Ths.Nguyễn Quang Vũ

Đà nẵng, tháng 7 năm 2011

LỜI CẢM ƠN

Như vậy là ba năm học tại trường Cao Đẳng Công Nghệ Thông Tin Hữu Nghị Việt Hàn sắp kết thúc. Đến lúc này em vẫn nghĩ mình thật may mắn khi được vào học tại trường Việt Hàn. Cơ sở vật chất của trường rất tốt, thầy cô giảng viên rất nhiệt tình, các thầy cô trong ban giám hiệu trường chăm lo tới sinh viên, tất cả điều đó đã để lại cho em những ấn tượng rất khó quên.

Đặc biệt em xin dành nhiều tình cảm đến các thầy cô trong nghành Hệ Thống Thông Tin, thầy Lê Viết Trương, thầy Hồ Văn Phi, cô Nguyễn Phương Tâm, cô Nguyễn Thị Hoa Huệ, thầy Nguyễn Văn Lợi. Trong suốt ba năm học tại trường, các thầy cô đã tận tình chăm lo, dẫn dắt chúng em trong học tập cũng như trong đời sống hằng ngày. Trước khi vào học tại trường, em chưa bao giờ nghĩ các thầy cô lại có thể quan tâm, thân thiết với sinh viên đến như thế. Trong thời gian học tập tại trường, chúng em đã rất nhiều lần làm cho các thầy cô buồn lòng, nhưng với tấm lòng yêu thương sinh viên, thầy cô đã bỏ qua cho chúng em, thật chúng em không biết phải báo đáp với thầy cô như thế nào nữa. Một lần nữa em xin cảm ơn các thầy cô, em xin hứa khi ra đời sẽ cố gắng sống xứng đáng với những gì mà các thầy cô đã dạy bảo em.

Báo cáo đồ án tốt nghiệp, đó chính là kỳ thi quan trọng nhất trong đời sinh viên, là kỳ thi thể hiện chứng tỏ mỗi sinh viên đã học được những gì trong suốt các năm học tại trường. Em sau ba năm nỗ lực, đã thật vinh dự và tự hào khi được tham dự kỳ báo cáo đồ án tốt nghiệp này. Trong suốt ba tháng, dựa vào sự nỗ lực của bản thân, được sự giúp đỡ của các thầy cô, em đã hoàn thành bản báo cáo đồ án tốt nghiệp chuyên ngành Hệ Thống Thông Tin. Em xin chân thành cảm ơn thầy Nguyễn Quang Vũ, thầy đã tận tình chỉ bảo, hướng dẫn em, giúp em hoàn thành đồ án của mình.

Một lần nữa, em xin cảm ơn tất cả các thầy cô, bạn bè đã chỉ bảo, giúp đỡ em về học tập cũng như đời sống. Em sẽ luôn nhớ về những điều đó như một kỷ niệm khó quên trong đời sinh viên của mình.

Nguyễn Trọng Trí

MỤC LỤC

DANH MỤC CÁC TỪ TIẾNG ANH

Class: lớp

Graphical User Interface (GUI): giao diện đồ họa người dùng

Object Oriented Programming (OOP): lập trình hướng đối tượng

GUI Component: thành phần đồ họa người dùng

Framework: nền tảng

Request: Yêu cầu

Server: máy chủ

Client: máy trạm

Control: đối tượng điều khiển

Test-driven development (TDD): phát triển điều hướng bởi kiểm thử

Unit test: kiểm thử đơn vị

Postback: phản hồi

Test: kiểm thử

Code: mã lệnh

ConnectionString : chuỗi kết nối

Language Integrate Query (LINQ): ngôn ngữ truy vấn tích hợp

Solution: các giải pháp cho dự án

Project: dự án

Browser : trình duyệt

Businness logic: lớp xử lý

Implement: thực thi các phương thức từ một lớp

Application Programming Interface - API: giao diện lập trình ứng dụng

Render: trả lại, biểu diễn

Override : nạp chồng

Redirect: chuyển hướng

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH

MỞ ĐẦU

- **Lý do chọn đề tài:**

Hiện nay việc thiết kế một trang web ASP.NET rất dễ dàng, chúng ta có thể tìm tài liệu trên mạng, tham khảo các project để nghiên cứu và xây dựng. Bởi vì do Microsoft muốn tạo ra một công cụ để người sử dụng có thể dễ dàng làm việc và xây dựng một trang web nhanh chóng nhất, ASP.NET WebForm được thiết kế để thực hiện những điều đó.

ASP.NET Webform được thiết kế để người dùng cảm thấy như mình đang thiết kế một chương trình Windows Form vậy, bằng cách kéo thả các button, tự sinh code HTML, đơn giản, dễ hiểu… Chính vì thế nền tảng ASP.NET WebForm dù đã ra đời cách đây hơn 10 năm nhưng hiện nay vẫn đang sử dụng rộng rãi.

Tuy nhiên ưu điểm của ASP.NET WebForm đôi khi lại chính là nhược điểm của nó, chính là không có sự phân chia rõ ràng giữa giao diện và code xử lý, nên ngay trong trang giao diện lại có câu lệnh truy vấn Sql. Chính cái tiện lợi là một tính năng nào đó được xây dựng thì trong đó có cả mã HTML, Css, Javascrip, lệnh xử lý sự kiện… Đến khi chúng ta cần thay thế hoặc nâng cấp một chức năng nào đó thì rất rắc rối. Để gọi là khắc phục những nhược điểm của ASP.NET Webform, năm 2007 Microsoft đã cho ra đời nền tảng ASP.NET MVC.

ASP.NET MVC là một lựa chọn thay thế cho ASP.NET WebForm, được xây dựng với 3 lớp chính, lớp giao diện (Views), lớp điều khiển (Controllers) và lớp dữ liệu (Models). Việc chia một trang web thành nhiều lớp như thế này giúp cho những lập trình viên có kinh nghiệm có thể xây dựng một website với cấu trúc chặt chẽ, rõ ràng. Với cấu trúc 3 lớp như thế này, việc nâng cấp hoặc thay thế một chức năng nào đó trở nên hết sức dễ dàng, đồng thời việc kiểm thử cũng trở nên đơn giản hơn.

Với những ưu điểm trên, trong tương lai chắc chắn ASP.NET MVC sẽ là một nền tảng chính trong việc xây dựng và phát triển một website ASP.NET. Tuy nhiên vì đây là một công nghệ mới, nên tại Việt Nam hầu như chưa được áp dụng nhiều, cũng có rất ít bạn sinh viên biết tới mô hình này. Chính vì thế, em đã mạnh dạn chọn đề tài nghiên cứu về MVC để làm đồ án tốt nghiệp của mình.

- **Mục đích nghiên cứu**

Em nghiên cứu ASP.NET MVC nhằm những mục đích sau:

- Học được những kiến thức mới.
- Trong quá trình tìm hiểu sẽ giúp em nâng cao khả năng tự học của mình.
- Phục vụ cho việc xây dựng trang website mua bán sách, nhằm mục đích là áp dụng được những gì đã học được đưa vào thực tế.
- Tạo nguồn tài liệu ASP.NET MVC tiếng việt cho những ai cần tìm hiểu.

- **Đối tượng và phạm vi nghiên cứu**

Nghiên cứu ASP.NET MVC 2.0, nghiên cứu các kiến thức liên quan.

- **Phương pháp nghiên cứu**

Nghiên cứu lý thuyết thông qua các ebook được phát hành bởi Microsoft. Tìm hiểu những ví dụ trên mạng, từng bước áp dụng vào các chương trình thử nghiệm. Sau đó tổng hợp lại kiến thức và hoàn thành báo cáo và sản phẩm demo.

- **Ý nghĩa khoa học và thực tiễn của đề tài**

- Hiện tại ASP.NET MVC là một công nghệ còn mới tại Việt Nam, có rất nhiều bạn sinh viên chưa được biết đến công nghệ này, đặc biệt là các bạn sinh viên tại trường Việt Hàn. Chính vì thế việc hoàn thành bản báo cáo và chương trình ứng dụng sẽ giúp ích rất nhiều cho các bạn sinh viên muốn tiếp cận công nghệ mới một cách nhanh nhất, tại vì hiện tại tài liệu tiếng việt cho ASP.NET MVC là rất hiếm, đồng thời có sẵn một ứng dụng demo sẽ giúp các bạn dễ hiểu và áp dụng hơn.
- ASP.NET MVC có rất nhiều ưa điểm vượt trội, em nghĩ tương lại ASP.NET MVC sẽ dần thay thế ASP.NET Webform trong việc xây dựng website. Cho nên việc tìm hiểu công nghệ này sẽ có tính thực tiễn rất cao, có thể áp dụng ngay bây giờ hoặc trong tương lai.

PHẦN 1

GIỚI THIỆU MÔ HÌNH MVC

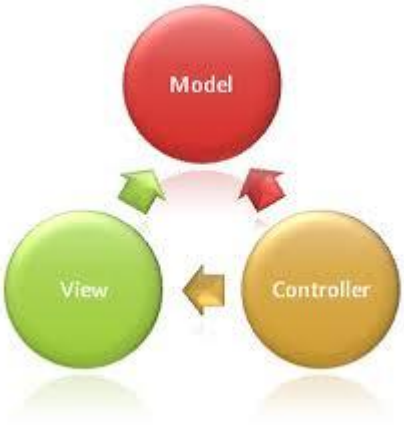
1.1. Xuất xứ

Tất cả bắt đầu vào những năm 70 của thế kỷ 20, tại phòng thí nghiệm Xerox PARC ở Palo Alto. Sự ra đời của giao diện đồ họa (Graphical User Interface) và lập trình hướng đối tượng (Object Oriented Programming) cho phép lập trình viên làm việc với những thành phần đồ họa như những đối tượng đồ họa có thuộc tính và phương thức riêng của nó. Không dừng lại ở đó, những nhà nghiên cứu ở Xerox PARC còn đi xa hơn khi cho ra đời cái gọi là kiến trúc MVC (viết tắt của Model – View – Controller).

MVC được phát minh tại Xerox Parc vào những năm 70, bởi TrygveReenskaug. MVC lần đầu tiên xuất hiện công khai là trong Smalltalk-80. Sau đó trong một thời gian dài hầu như không có thông tin nào về MVC, ngay cả trong tài liệu 80 Smalltalk. Các giấy tờ quan trọng đầu tiên được công bố trên MVC là "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk - 80", bởi Glenn Krasner và Stephen Pope, xuất bản trong tháng 8 / tháng 9 năm 1988.

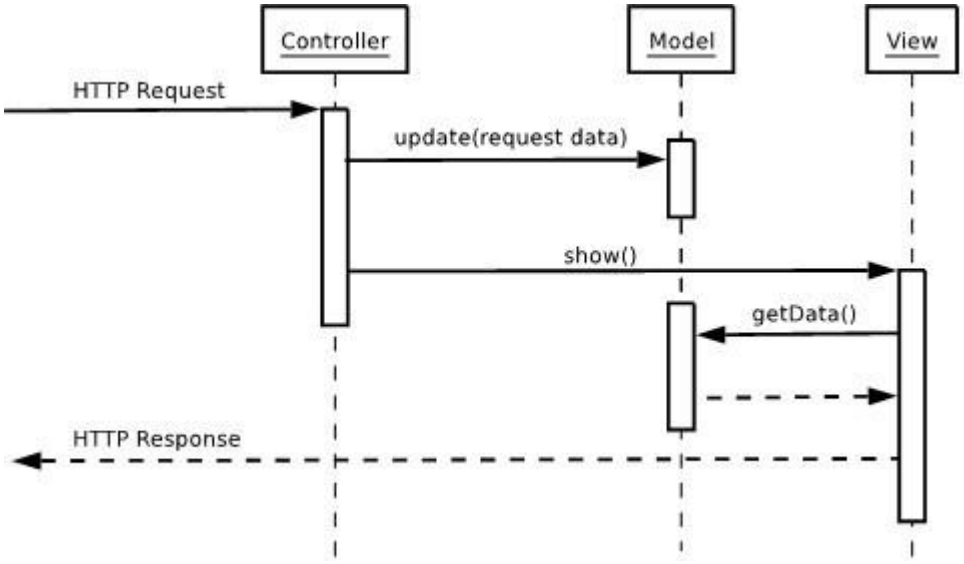
1.2. Kiến trúc của mô hình MVC

Trong kiến trúc MVC, một đối tượng đồ họa người dùng (GUI Component) bao gồm 3 thành phần cơ bản: Model, View, và Controller. Model có trách nhiệm đối với toàn bộ dữ liệu cũng như trạng thái của đối tượng đồ họa. View chính là thể hiện trực quan của Model, hay nói cách khác chính là giao diện của đối tượng đồ họa. Và Controller điều khiển việc tương tác giữa đối tượng đồ họa với người sử dụng cũng như những đối tượng khác.



Hình 1.1: Các thành phần chính của mô hình MVC

Khi người sử dụng hoặc những đối tượng khác cần thay đổi trạng thái của đối tượng đồ họa, nó sẽ tương tác thông qua Controller của đối tượng đồ họa. Controller sẽ thực hiện việc thay đổi trên Model. Khi có bất kỳ sự thay đổi nào ở xảy ra ở Model, nó sẽ phát thông điệp (broadcast message) thông báo cho View và Controller biết. Nhận được thông điệp từ Model, View sẽ cập nhật lại thể hiện của mình, đảm bảo rằng nó luôn là thể hiện trực quan chính xác của Model. Còn Controller, khi nhận được thông điệp từ Model, sẽ có những tương tác cần thiết phản hồi lại người sử dụng hoặc các đối tượng khác.



Hình 3.2: Mô hình tuần tự của MVC

Ví dụ:

Lấy ví dụ một GUI Component (thành phần đồ họa người dùng) đơn giản là Checkbox. Checkbox có thành phần Model để quản lý trạng thái của nó là check hay uncheck, thành phần View để thể hiện nó với trạng thái tương ứng lên màn hình, và thành phần Controller để xử lý những sự kiện khi có sự tương tác của người sử dụng hoặc các đối tượng khác lên Checkbox.

Khi người sử dụng nhấn chuột vào Checkbox, thành phần Controller của Checkbox sẽ xử lý sự kiện này, yêu cầu thành phần Model thay đổi dữ liệu trạng thái. Sau khi thay đổi trạng thái, thành phần Model phát thông điệp đến thành phần View và Controller. Thành phần View của Checkbox nhận được thông điệp sẽ cập nhật lại thể hiện của Checkbox, phản ánh chính xác trạng thái Checkbox do Model lưu giữ. Thành phần Controller nhận được thông điệp do Model gởi tới sẽ có những tương tác phản hồi với người sử dụng nếu cần thiết.

1.3. Đặc điểm của mô hình MVC

Cái lợi ích quan trọng nhất của mô hình MVC là nó giúp cho ứng dụng dễ bảo trì, module hóa các chức năng, và được xây dựng nhanh chóng. MVC tách các tác vụ của ứng dụng thành các phần riêng lẽ model, view, controller giúp cho việc xây dựng ứng dụng nhẹ nhàng hơn. Dễ dàng thêm các tính năng mới, và các tính năng cũ có thể dễ dàng thay đổi. MVC cho phép các nhà phát triển và các nhà thiết kế có thể làm việc đồng thời với nhau. MVC cho phép thay đổi trong 1 phần của ứng dụng mà không ảnh hưởng đến các phần khác.

Sở dĩ như vậy vì kiến trúc MVC đã tách biệt (decoupling) sự phụ thuộc giữa các thành phần trong một đối tượng đồ họa, làm tăng tính linh động (flexibility) và tính tái sử dụng (reusebility) của đối tượng đồ họa đó. Một đối tượng đồ họa bấy giờ có thể dễ dàng thay đổi giao diện bằng cách thay đổi thành phần View của nó trong khi cách thức lưu trữ (Model) cũng như xử lý (Controller) không hề thay đổi.

Tương tự, ta có thể thay đổi cách thức lưu trữ (Model) hoặc xử lý (Controller) của đối tượng đồ họa mà những thành phần còn lại vẫn giữ nguyên.

Chính vì vậy mà kiến trúc MVC đã được ứng dụng để xây dựng rất nhiều framework và thư viện đồ họa khác nhau. Tiêu biểu là bộ thư viện đồ họa của ngôn ngữ lập trình hướng đối tượng SmallTalk (cũng do Xerox PARC nghiên cứu và phát triển vào thập niên 70 của thế kỷ 20). Các Swing Components của Java cũng được xây dựng dựa trên kiến trúc MVC. Đặc biệt là nền tảng ASP.NET MVC Framework sẽ được em trình bày ở chương sau đây.

PHẦN 2

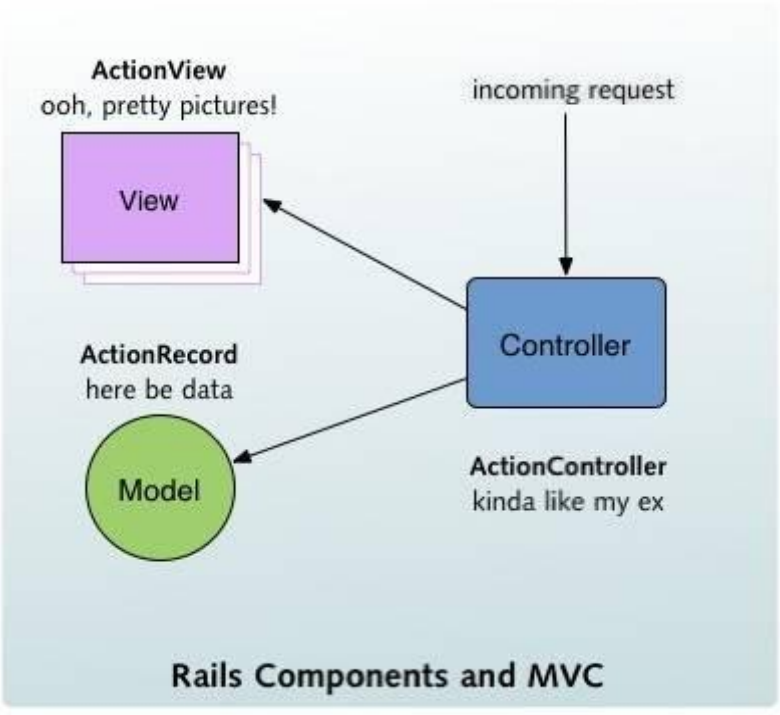
MÔ HÌNH MVC TRONG ASP.NET

2.1. Giới thiệu tổng quan

Như đã nói ở phần 1, mô hình MVC với những ưu điểm đã được ứng dụng nhiều trên các nền tảng (framework) khác nhau, trong đó có một nền tảng (framework) nổi tiếng được nhiều người biết đến và sử dụng đó là nền tảng (framework) ASP.NET MVC. Vậy ASP.NET MVC là gì ?

ASP.NET MVC là một nền tảng (framework) phát triển ứng dụng web mới của Microsoft, nó kết hợp giữa tính hiệu quả và nhỏ gọn của mô hình model-view-controller(MVC), những ý tưởng và công nghệ hiện đại nhất, cùng với những thành phần tốt nhất của nền tảng ASP.NET hiện thời. Là một lựa chọn khác bên cạnh nền tảng WebForm khi phát triển 1 ứng dụng web sử dụng ASP.NET.

Trong chương này em sẽ trình bày lý do lại sao ASP.NET MVC được tạo ra, nó có gì khác so với ASP.NET WebForm, và cuối cùng là những cái mới trong ASP.NET MVC.



Hình 4.1: Mô hình MVC

2.1.1. Lịch sử phát triển của ASP.NET

Để hiểu được những đặc điểm nổi bật và mục tiêu thiết kế của ASP.NET MVC, trước tiên cần coi lại lịch sử phát triển của website ASP. Trong số những nền tảng phát triển web của Microsoft, chúng ta sẽ thấy sức mạnh và sự phức tạp tăng lên theo từng năm. Như trong bảng ta thấy mỗi nền tảng mới đều giải quyết những thiếu sót đặc trưng của nền tảng trước đó. Tương tự, ASP.NET MVC được thiết kế để giải quyết những thiếu sót của ASP.NET WebForms truyền thống, nhưng lại bằng cách nhấn mạnh sự đơn giản.

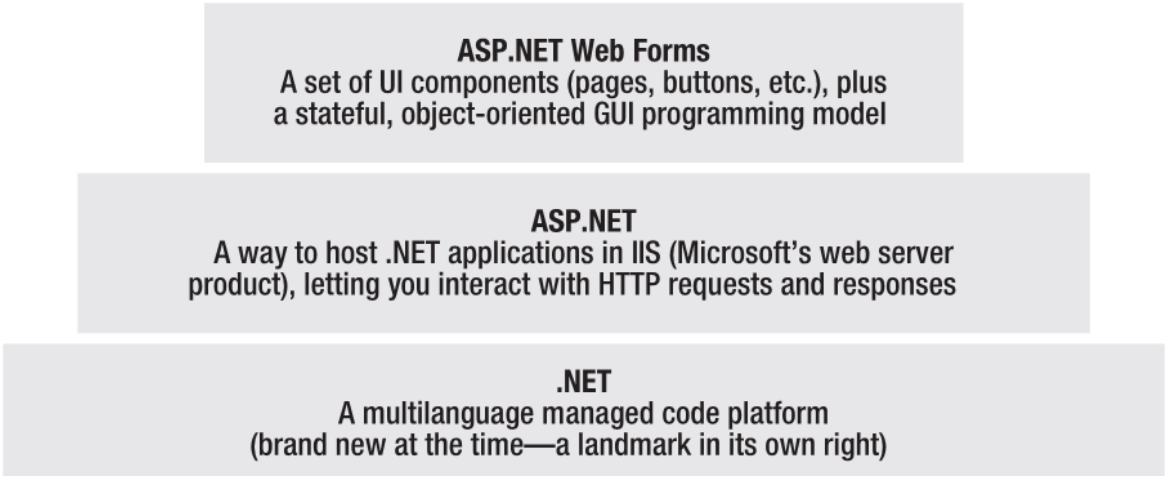
Bảng 2.1: Quá trình phát triển của Asp.Net

Thời kì	Công nghệ	Sức mạnh	Điểm yếu
Thời kỳ đầu	Common Gateway Interface (CGI)	Đơn giản, linh hoạt Lựa chọn duy nhất vào thời điểm này	Chạy ngòai web server, nên cần nhiều tài nguyên (cần các tiến trình riêng lẻ trên HĐH cho mỗi lời yêu cầu) Cấp thấp
Thời kỳ tiếp theo	Microsoft Internet Database Connector (IDC)	Chạy trong web server	Chỉ là đóng gói cho những câu truy vấn SQL và template cho các kết quả có định dạng
1996	Active Server Pages (ASP)	Mục đích chung	Thông dịch thời gian thực Xu hướng "spaghetti code"
2002/03	ASP.NET 1.0/1.1	Đã được biên dịch Giao diện có trạng thái	Nặng nề trên băng thông HTML khó nhìn

		Cấu trúc lớn Xu hướng lập trình hướng đối tượng	Không thể test
2005	ASP.NET 2.0		
2007	ASP.NET Ajax		
2008	ASP.NET 3.5		
2009	ASP.NET MVC 1.0		
2010	ASP.NET MVC 2.0		
2011	ASP.NET MVC 3.0		

- ASP.NET truyền thống:

ASP.NET đã là 1 bước nhảy vọt khi lần đầu tiên xuất hiện, nhằm thu hẹp khoảng cách giữa phát triển Window Form hướng đối tượng (có trạng thái) và phát triển web hướng HTML (không có trạng thái). Hình dưới minh họa công nghệ Asp.net WebForm trong lần đầu xuất hiện năm 2002.



Hình 2.2: Mô hình ASP.NET Webform

Microsoft đã cố gắng che dấu HTTP (không trạng thái) và HTML (vào thời điểm đó không thân thiện với nhiều người lập trình) bằng cách dùng mô hình giao diện như một đối tượng điều khiển (control) có cấu trúc hoạt động phía server.

Mỗi đối tượng điều khiển (control) lưu giữ trạng thái qua các yêu cầu (request) (sử dụng tính năng ViewState), tự động tạo ra mã HTML khi cần thiết, và tự động kết nối với các sự kiện phía client (ví dụ như click) với mã hồi đáp phía server. Kết quả WebForm là một lớp trừu tượng lớn nhằm chuyển giao diện có xử lý sự kiện thông qua Web.

Nhược điểm của ASP.NET

ASP.NET truyền thống đã từng là một ý tưởng hay, nhưng thực tế lại trở nên rắc rối. Qua nhiều năm, sử dụng ASP.NET WebForm cho thấy có một số nhược điểm:

- ViewState (trạng thái hiển thị): Kĩ thuật lưu giữ trạng thái qua các yêu cầu (request) thường mang lại kết quả là những khối dữ liệu lớn được chuyển qua lại giữa client và server. Nó có thể đạt hàng trăm kilobytes trong nhiều dữ liệu thực, và nó đi qua đi lại với mỗi lần yêu cầu (request), làm những người truy cập vào trang web phải chờ một thời gian dài khi họ click một button hoặc cố gắng di chuyển đến trang kế tiếp. ASP.NET bị tình trạng này rất tồi tệ, Ajax là một trong các giải pháp được đưa ra để giải quyết vấn đề này.
- Page life cycle (chu kỳ sống của một trang web): Kĩ thuật kết nối sự kiện phía client với mã xử lý sự kiện phía server là một phần của page life cycle, có thể cực kì rắc rối và mỏng manh. Chỉ có một số ít lập trình viên thành công trong việc xử lý hệ thống đối tượng điều khiển (control) trong thời gian thực mà không bị lỗi ViewState hoặc hiểu được rằng một số trình xử lý sự kiện không được kích hoạt một cách bí hiểm.
- Limited control over HTML (giới hạn kiểm soát HTML): Server control tự tạo ra nó như là mã HTML, nhưng không phải là mã HTML mà bạn muốn. Ngoài việc mã HTML của chúng thường không tuân theo tiêu chuẩn web hoặc không sử dụng tốt CSS mà hệ thống các server control còn tạo ra các giá trị ID phức tạp và không đoán trước dc, làm khó khăn trong việc sử dụng JavaScript.
- Ý thức sai về sự tách biệt các thành phần: Mô hình code-behind của ASP.NET cung cấp một giải pháp cho phép ứng dụng đưa mã ra khỏi các dòng HTML vào thành một lớp code – behind riêng biệt. Điều này đã được ca ngợi là làm

tách biệt giữa giao diện với mã xử lý, nhưng thực tế người lập trình được khuyến khích pha trộn mã xử lý giao diện (xử lý cây control phía server) với mã xử lý chương trình (xử lý CSDL) trong cùng những lớp code behind khổng lồ. Nếu không có sự tách biệt rõ ràng giữa các thành phần, kết quả cuối cùng thường là mỏng manh và khó hiểu.

- Untestable (kiểm chứng): Khi những người thiết kế của ASP.NET lần đầu tiên giới thiệu nền tảng này, họ có thể đã không lường trước được là việc kiểm thử (test) tự động sẽ trở thành một công đoạn chính của việc phát triển phần mềm ngày nay. Không quá ngạc nhiên, cấu trúc mà họ đã thiết kế hoàn toàn không thích hợp với việc kiểm thử (test) tự động.

ASP.NET tiếp tục bổ sung thêm các tính năng. Phiên bản 2.0 thêm nhiều thành phần (component) chuẩn có thể giảm khá nhiều lệnh mà bạn cần phải tự viết. AJAX release năm 2007 đã là phản hồi của Microsoft với phong trào Web 2.0/Ajax hồi đó, hỗ trợ tương tác phía client trong khi làm cho công việc của người lập trình đơn giản hơn. Phiên bản 3.5 là một bản nâng cấp nhỏ hơn, thêm hỗ trợ cho các tính năng của .NET 3.5 và các đối tượng điều khiển (control) mới. Tính năng ASP.NET Dynamic Data tạo ra các trang cho phép chỉnh sửa / liệt kê cơ sở dữ liệu một cách tự động .

➢ Sự ra đời của ASP.NET MVC:

Vào tháng 11 năm 2007, ở hội thảo ALT.NET tại Austine, Texas, giám đốc điều hành Scott Guthrie của Microsoft đã công bố và mô tả về nền tảng phát triển web MVC mới, xây dựng trên ASP.NET, rõ ràng được thiết kế như là một lời hồi đáp thẳng vào các lời chỉ trích trước đây.

ASP.NET MVC cung cấp sự tách biệt rõ ràng giữa các thành phần bất chấp việc sử dụng lại mô hình MVC mặc dù nó không có gì mới – MVC lần đầu được công bố vào năm 1978 trong dự án Smalltalk của Xerox PARC - nhưng ngày nay nó phổ biến như là một kiến trúc cho các ứng dụng web bởi vì các lý do sau :

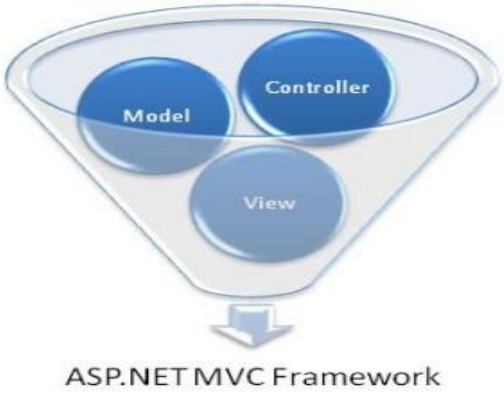
- Người dùng tương tác với ứng dụng MVC tự nhiên sẽ theo một chu trình : người dùng thực hiện một hành động, và để hồi đáp lại, ứng dụng thay đổi mô hình dữ liệu của nó và chuyển một trang đã được cập nhật cho người dùng, và sau đó

vòng xoay lập lại. Điều này là thích hợp cho một ứng dụng web phải luân chuyển hàng loạt các lời yêu cầu cũng như hồi đáp.

- Những ứng dụng web luôn cần phải kết hợp một số công nghệ (về cơ sở dữ liệu, HTML và mã xử lý), luôn chia thành nhiều lớp, và những mẫu đề ra đã nảy sinh thành các khái niệm trong MVC. ASP.NET MVC thực thi một biến thể hiện đại trên MVC mà đặc biệt thích hợp với các ứng dụng web.

2.1.2. Khái quát các thành phần của ASP.NET MVC

Như đã giới thiệu ở chương trên, ASP.NET MVC cũng chia nhỏ một ứng dụng thành ba thành phần để cài đặt, mỗi thành phần đóng một vai trò khác nhau và ảnh hưởng lẫn nhau, đó là models, views, và controllers.



Hình 2.3: Nền tảng Asp.Net MVC Framwork

Models trong các ứng dụng dựa trên MVC là những thành phần có nhiệm vụ lưu trữ thông tin, trạng thái của các đối tượng, thông thường nó là một lớp được ánh xạ từ một bảng trong CSDL. Lấy ví dụ, chúng ta có lớp Giáo trình được sử dụng để mô tả dữ liệu từ bảng Giáo trình trong SQL, bao gồm Mã giáo trình, Tên giáo trình...

Views chính là các thành phần chịu trách nhiệm hiển thị các thông tin lên cho người dùng thông qua giao diện. Thông thường, các thông tin cần hiển thị được lấy từ thành phần Models.

Ví dụ, đối tượng Giáo trình có một "Edit" view bao gồm các textbox, các dropdown và checkbox để chỉnh sửa các thuộc tính của thông tin giáo trình; có một "Display" view gồm 2 dòng, cột dòng là Mã giáo trình, dòng sau là Tên giáo trình... để xem thông tin về sinh viên.

Controllers trong các ứng dụng kiểu MVC chịu trách nhiệm xử lý các tác động về mặt giao diện, các thao tác đối với models, và cuối cùng là chọn một view thích hợp để hiển thị ra màn hình. Trong kiến trúc MVC, View chỉ có tác dụng hiển thị giao diện mà thôi, còn điều kiển dòng nhập xuất của người dùng vẫn do Controllers đảm trách.

2.1.3. Lợi ích của mô hình ASP.NET MVC

- Có tính mở rộng do có thể thay thế từng thành phần 1 cách dễ dàng
- Không sử dụng Viewstate, điều này làm các nhà phát triển dễ dàng điều khiển ứng dụng của mình.
- Hệ thống định tuyến mới mạnh mẽ.
- Hỗ trợ tốt hơn cho test-driven development (TDD – mô hình phát triển kiểm thử) cài đặt các kiểm thử đơn vị (unit tests) tự động, xác định và kiểm tra lại các yêu cầu trước khi bắt tay vào viết code.
- Hỗ trợ kết hợp rất tốt giữa người lập trình và người thiết kế giao diện.
- Sử dụng các tính năng tốt nhất đã có của ASP.NET.

2.1.4. So sánh ASP.NET MVC với ASP.NET

Bạn đã được nghe qua về điểm yếu và giới hạn của ASP.NET WebForm truyền thống, và làm thế nào mà ASP.NET MVC vượt qua những vấn đề này. Điều đó không có nghĩa là ASP.NET WebForm đã chết mà chỉ là : Microsoft muốn mọi người hiểu rằng có hai nền tảng song song nhau, hỗ trợ cho nhau, và cả hai đều là đối tượng cho việc phát triển hiện tại. Nói chung, việc bạn chọn lựa giữa hai mô hình còn tùy vào hoàn cảnh.

ASP.NET WebForm mang tới một trang web mà giao diện có thể lưu giữ trạng thái, và cuối cùng thêm vào một lớp trừu tượng tinh vi nằm trên HTTP và HTML, sử dụng ViewState và postback để tạo ra hiệu ứng của việc có trạng thái . Điều này thích hợp với phong cách phát triển kéo và thả của Window Form, tức là bạn đặt các đối tượng có giao diện lên trang và mã xử lý vào trình xử lý sự kiện của chúng.

MVC hòa vào bản chất không trạng thái của HTTP, làm việc chung với nó hơn là chống lại. Điều này yêu cầu bạn phải hiểu thật sự cách làm việc của một ứng dụng web, để đạt được điều đó, MVC cung cấp một cách tiếp cận đơn giản, mạnh mẽ và hiện đại cho việc viết các ứng dụng web với mã có trật tự mà dễ dàng để kiểm thử (test) và bảo trì sau này, giải phóng những phức tạp khó chịu và các giới hạn không đáng có.

Điều đặc biệt là ASP.NET MVC có mã nguồn mở, không giống các nền tảng trước đó, bạn có thể dễ dàng tải mã nguồn gốc của ASP.NET MVC, thậm chí bạn có thể sửa đổi và tạo ra phiên bản của riêng bạn.

Có những tình huống mà ASP.NET WebForm khá tốt thậm chí còn tốt hơn ASP.NET MVC. Ví dụ như các ứng dụng nhỏ, nội bộ mà trực tiếp kết nối thẳng vào các bảng CSDL hoặc dẫn người sử dụng thông qua các trình hướng dẫn tự động (wizard). Vì thế sẽ không cần phải lo lắng về băng thông do ViewState, không dính dáng tới vấn đề tối ưu hóa hệ thống tìm kiếm, và không bị làm phiền về việc kiểm thử (test) và bảo trì lâu dài. Sự tiện lợi của cách phát triển kiểu kéo thả của ASP.NET WebForm làm mờ đi các điểm yếu của nó.

Nhưng mặt khác, nếu bạn viết 1 ứng dụng trên Internet, hoặc các ứng dụng nội bộ lớn hơn, bạn sẽ hướng tới tốc độ download nhanh và tương thích trình duyệt chéo, được xây dựng với chất lượng cao hơn, mã kiến trúc tốt thích hợp cho việc test tự động, trong trường hợp đó ASP.NET MVC sẽ mang lại nhưng ưu điểm quan trọng.

Bảng 2.2: So sánh giữa ASP.NET Webform và ASP.NET MVC

Tính năng	ASP.NET	ASP.NET MVC
Kiến trúc chương trình	Kiến trúc mô hình WebForm->Business->Database	Kiến trúc sử dụng việc phân chia chương trình thành Controllers, Models, View
Cú pháp chương trình	Sử dụng cú pháp WebForm, tất cả các sự	Các sự kiện được kiểu khiển bởi controllers, các

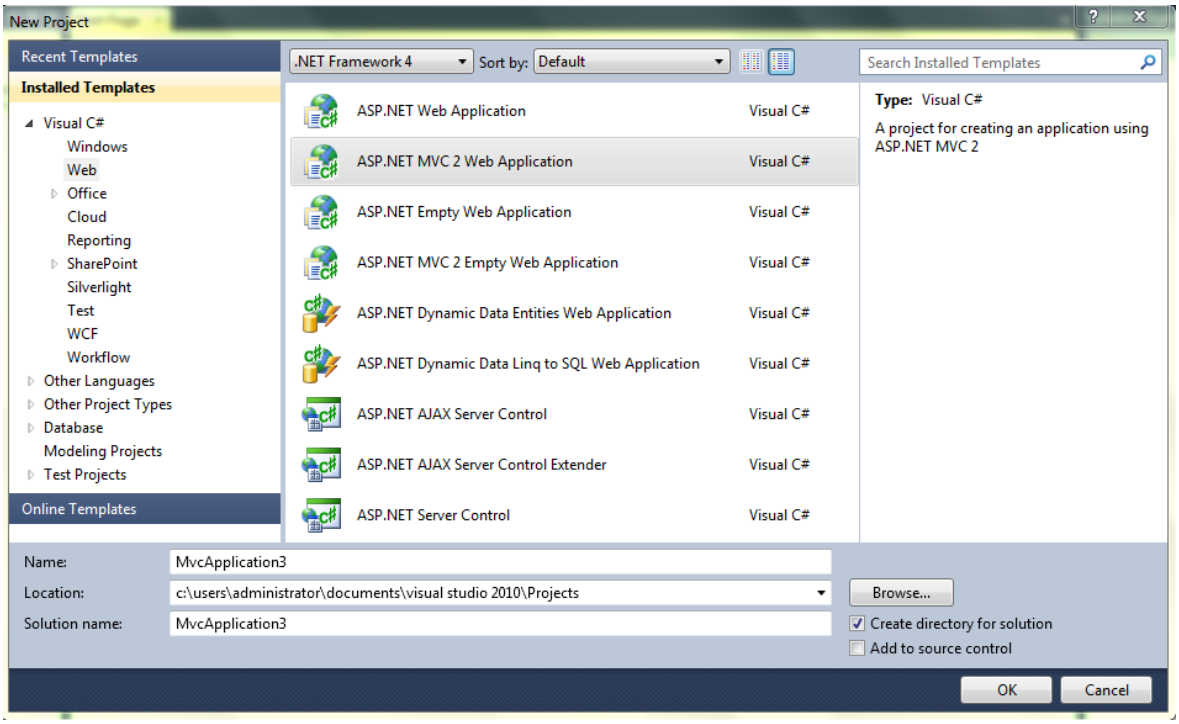
	kiện và control do server quản lý	control không do server quản lý
Truy cập dữ liệu	Sử dụng hầu hết các công nghệ truy cập dữ liệu trong ứng dụng	Phần lớn dùng LINQ to SQL class để tạo mô hình truy cập đối tượng
Debug	Debug chương trình phải thực hiện tất cả bao gồm các lớp truy cập dữ liệu, sự hiển thị, điều khiển các controls	Debug có thể sử dụng các unit test kiểm tra các phương thức trong controllers
Tốc độ phân tải	Tốc độ phân tải chậm trong khi trang có quá nhiều các controls vì ViewState quá lớn	Phân tải nhanh hơn do không phải quản lý ViewState để quản lý các control trong trang
Tương tác với javascrip	Tương tác với javascrip khó khăn vì các controls được điều khiển bởi server	Tương tác với javascrip dễ dàng vì các đối tượng không do server quản lý điều khiển không khó
URL Address	Cấu trúc địa chỉ URL có dạng <filename>.aspx?&<các tham số>	Cấu trúc địa chỉ rành mạch theo dạng Controllers/Action/ID

2.2. Cài đặt

Trước hết để tạo một ứng dụng ASP.NET MVC bạn cần phải đáp ứng các điều kiện sau:

- Do chúng ta cần phải cài bộ Visual Studio 2008 trở lên nên máy tính cần phải có cấu hình tốt thiểu: còn trống 5Gb ổ cứng, RAM 1,5G, chip phải đủ mạnh.
- Sau đó chúng ta cần phải cài đặt bộ Visual Studio 2008 hoặc 2010 (trong đồ án này em sử dụng Visual Studio 2010).

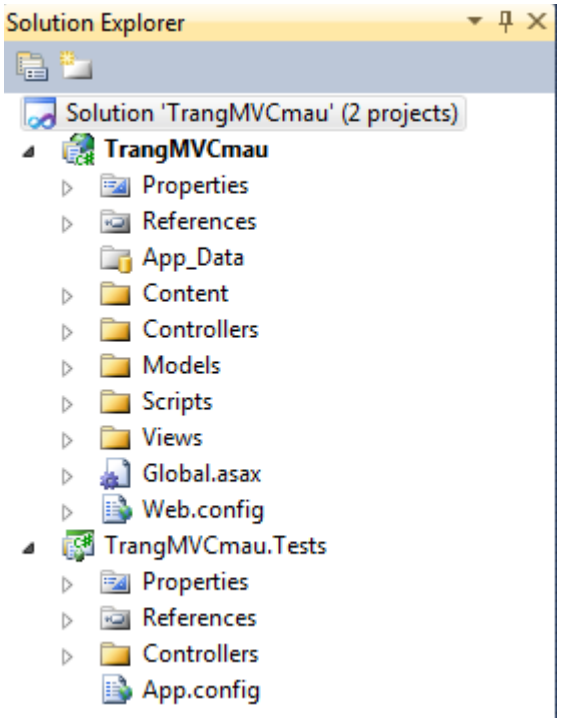
Sau khi chúng đã cài đặt xong bộ Visual Studio, để khởi tạo một dự án MVC, ta chọn File -> New Project (hoặc sử dụng phím tắt Ctrl + Shift + N). Ta có thể tùy chọn ngôn ngữ Visual Basic hoặc c# (trong đồ án này em chọn C#), ta chọn tiếp ứng dụng Web, chọn tiếp ASP.NET MVC 2 Web application.



Hình 2.4: Giao diện tạo project mới

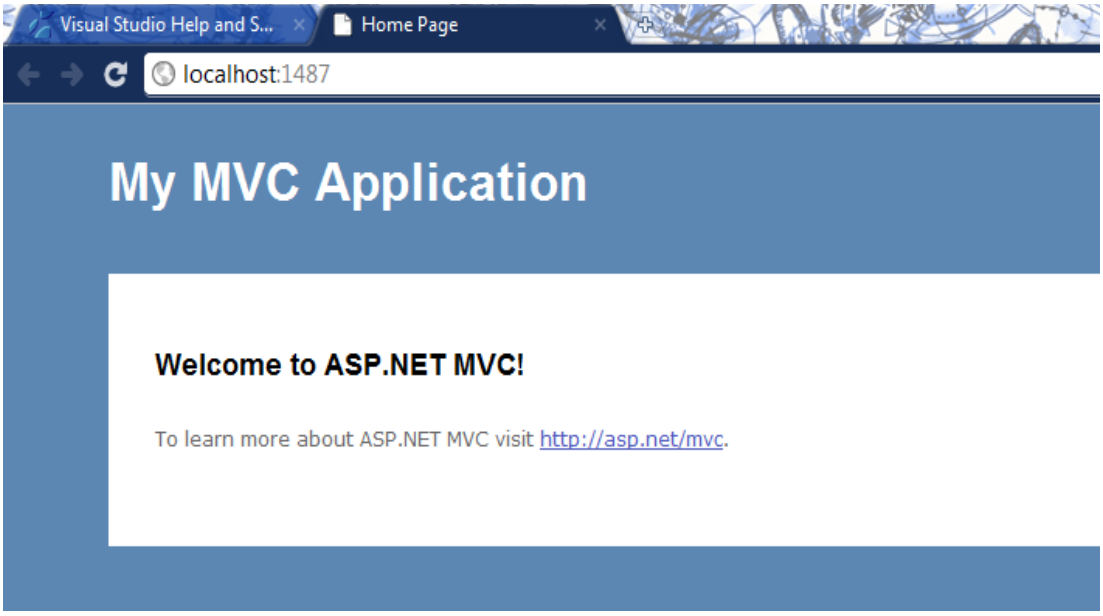
Ta có thể đặt lại tên dự án hoặc nơi lưu tùy ý chúng ta.

Ta nhấn button Ok, chương trình sẽ tạo cho chúng ta một ứng dụng Web MVC mẫu như sau:



Hình 2.5: Giao diện Solution của MVC

Bạn nhấn F5 để chạy chương trình.



Hình 2.6: Giao diện website ứng dụng mô hình MVC

Như vậy là chúng ta đã tạo cho mình một trang Web nhỏ ứng dụng mô hình MVC, trong các phần sau chúng ta sẽ tìm hiểu rõ hơn cách thức hoạt động của mô hình MVC trong ASP.NET.

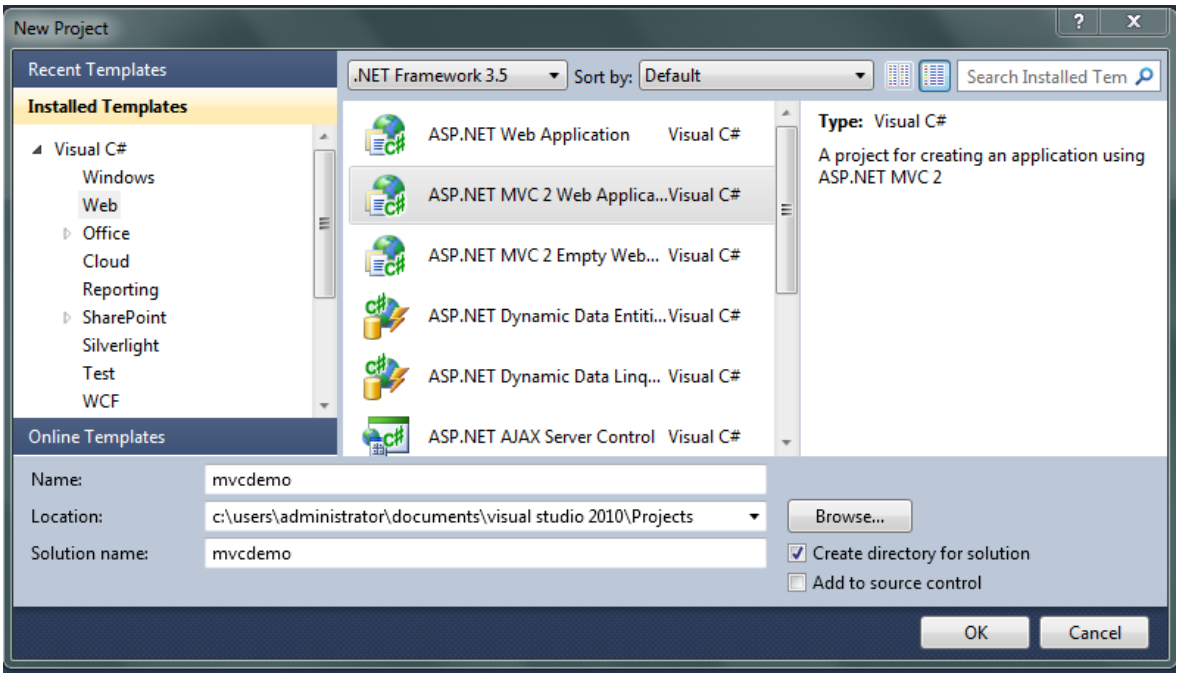
PHẦN 3

XÂY DỰNG ỨNG DỤNG VỚI ASP.NET MVC FRAMWORK

3.1. Tạo một project với ASP.NET MVC

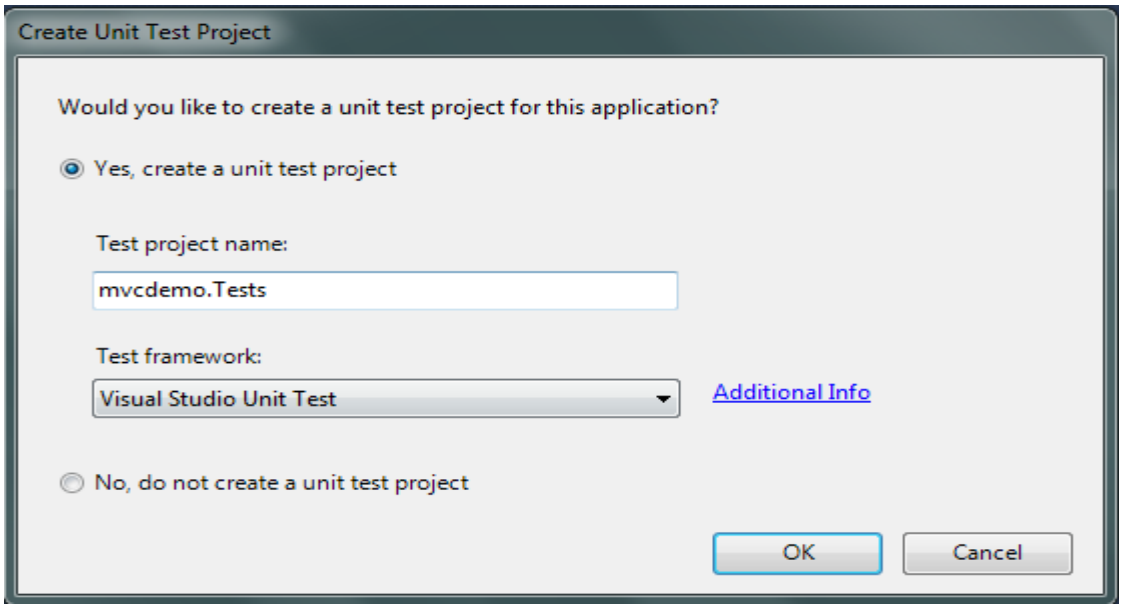
Trong bộ phần mềm Visual Studio 2010 đã có sẵn ASP.NET MVC FrameWork, ta tạo theo đường dẫn sau:

File -> New Project -> Visual C# -> Web -> ASP.NET MVC Web Application



Hình 3.1: Giao diện tạo project MVC

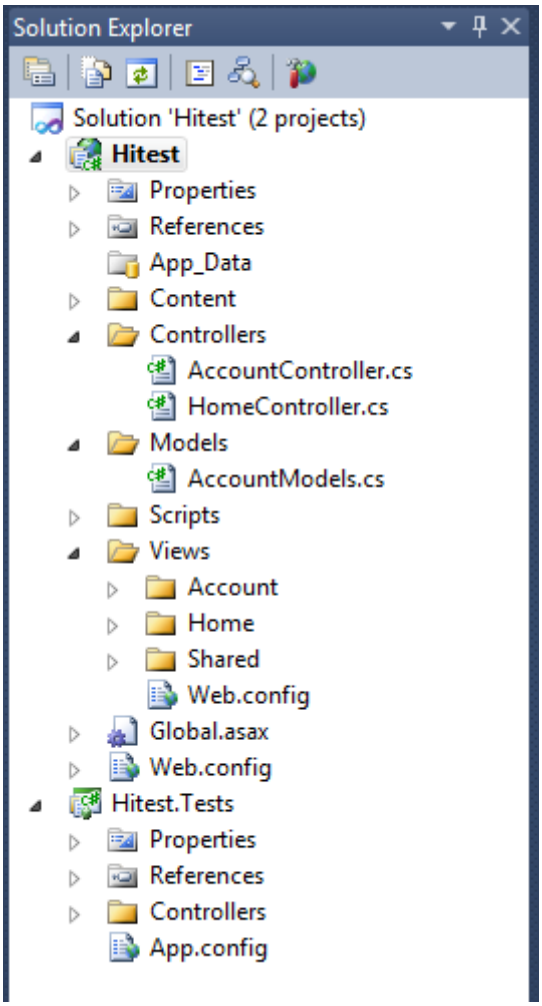
Khi tạo ASP.NET MVC Web application thì một hộp thoại Unit Test xuất hiện. Chọn Yes nếu muốn tạo một Project Test, ngược lại thì chọn No.



Hình 3.2: Thông báo hỏi có cho phép tạo Unit Test

Sau khi một ASP.NET MVC Web application được tạo, nhìn vào Solution Explore sẽ thấy 3 thư mục xuất hiện: Model, View, Controllers chứa các đối tượng tương ứng với các thành phần Models, View, Controllers trong mô hình MVC.

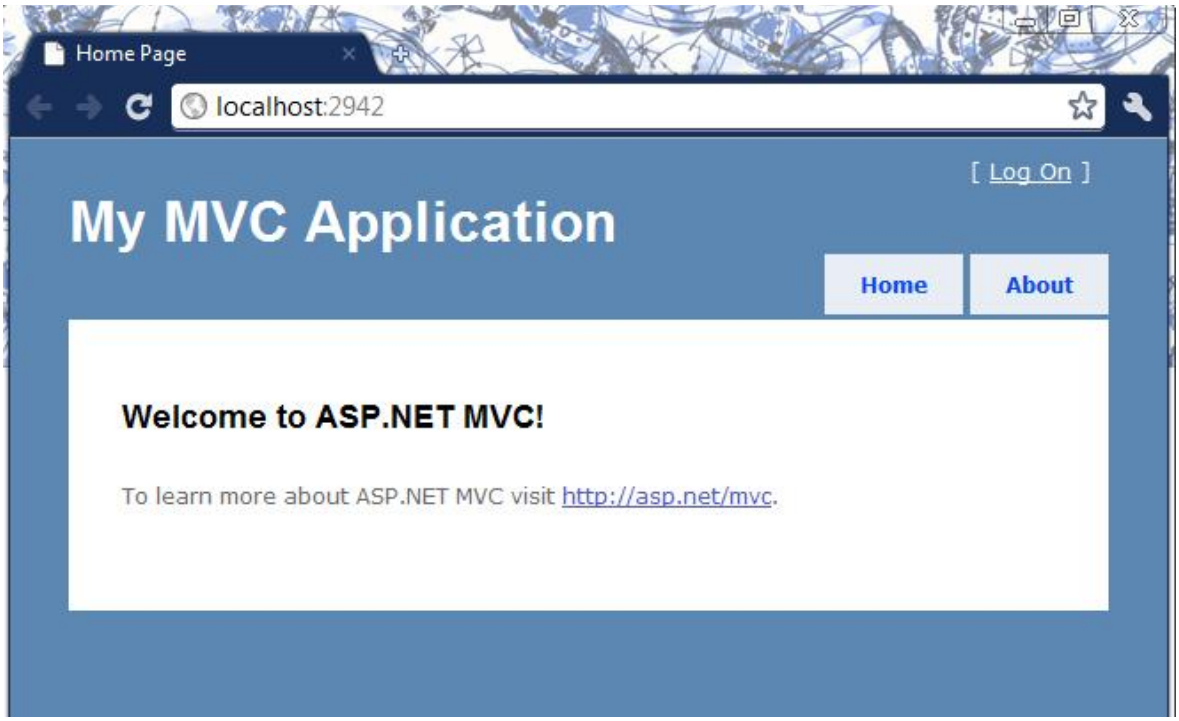
Visual Studio sẽ tạo một solution mới bao gồm hai projects con. Project đầu tiên là web project nơi mà ta sẽ cài đặt ứng dụng. Project thứ hai là testing project mà ta dùng để test project thứ nhất:



Hình 3.3: Giao diện Solution của MVC

Chạy ứng dụng bằng cách nhấn F5. Nếu là ứng dụng chạy lần đầu thì sẽ có thông báo hỏi có cho mở ở chế độ Debug hay không ? Nếu đồng ý chọn "Modify the Web.config file to enable debugging", không muốn Debug chọn "Run without debugging".

Kết quả khi chạy ứng dụng:



Hình 3.4: Giao diện website ứng dụng mô hình MVC

Cấu trúc thư mục trong Project:

Cấu trúc thư mục mặc định của ứng dụng ASP.MVC gồm có 3 thư mục chính:

- Controllers
- Models
- Views

ASP.NET MVC khuyến khích việc đưa những lớp (class) điều khiển vào bên trong thư mục /Controllers, những lớp (class) thuộc về mô hình dữ liệu vào bên trong thư mục /Models, và những gì liên quan đến giao diện vào thư mục /Views.

Mặc dù ASP.NET MVC Framework không bắt buộc chúng ta phải sử dụng cấu trúc này, nhưng đây là cấu trúc mặc định khi chúng ta tạo một dự án (project) mới và ASP.NET MVC luôn luôn khuyến khích việc sử dụng nó để phân chia ứng dụng. Ngoại trừ trường hợp ta đề ra một lý do đủ thuyết phục để thay đổi nó.

3.2. Tìm hiểu định tuyến URL

ASP.NET MVC Framework có một bộ máy ánh xạ URL thật sự mạnh mẽ. Bộ máy này cung cấp phương pháp rất linh hoạt trong việc ánh xạ URLs sang cho Controller. Bạn có thể dễ dàng định ra các quy luật ánh xạ, cài đặt để ASP.NET MVC dựa vào các quy luật ánh xạ đó, xác định xem phải thực thi Controller nào. ASP.NET MVC còn có khả năng phân tích URL, chuyển các thông số trong URL thành các tham số trong phần gọi hàm của Controller .

Đường đi mặc định từ ASP.NET MVC URL đến Controller Classes

Browser (trình duyệt) yêu cầu một địa chỉ từ Controller Acction trong ASP.NET MVC Framework được gọi là định tuyến URL (URL routing). URL routing sẽ chỉ định yêu cầu (request) tới Controller Action. URL routing sử dụng một bảng định tuyến để điều khiển các yêu cầu (request). Bảng định tuyến được tạo khi ứng dụng được chạy lần đầu tiên. Bảng định tuyến được thiết lập trong file Global.asax.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;
 namespace Totnghiep
{
   public class MvcApplication : System.Web.HttpApplication
   {
      public static void RegisterRoutes(RouteCollection routes)
      {
         routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
         routes.MapRoute(
            "Default", // Route name
```

```
            "{controller}/{action}/{id}", // URL with parameters
            new { controller = "Home", action = "Index", id = UrlParameter.Optional
} // Parameter defaults
        );
    }
    protected void Application_Start()
    {
        AreaRegistration.RegisterAllAreas();
        RegisterRoutes(RouteTable.Routes);
    }
  }
}
```

Khi ứng dụng chạy lần đầu tiên, phương thức Application_Start() được gọi. Phương thức này gọi một phương thức khác RegisterRouter(RouteTable.Router) để tạo bảng định tuyến.

Định tuyến mặc định chia một yêu cầu (request) thành 3 đoạn, mỗi phân đoạn nằm giữa 2 dấu "/". Phân đoạn đầu tiên chứa một Controller, phân đoạn thứ 2 chứa Controller Action, phân đoạn thứ 3 là tham số đầu vào của Controller Action.

Ví dụ: với địa chỉ **/Quanly/Editgiaotrinh/1** được hiểu là

```
Controller = Quanly
Action = Editgiaotrinh
Id = 1
```

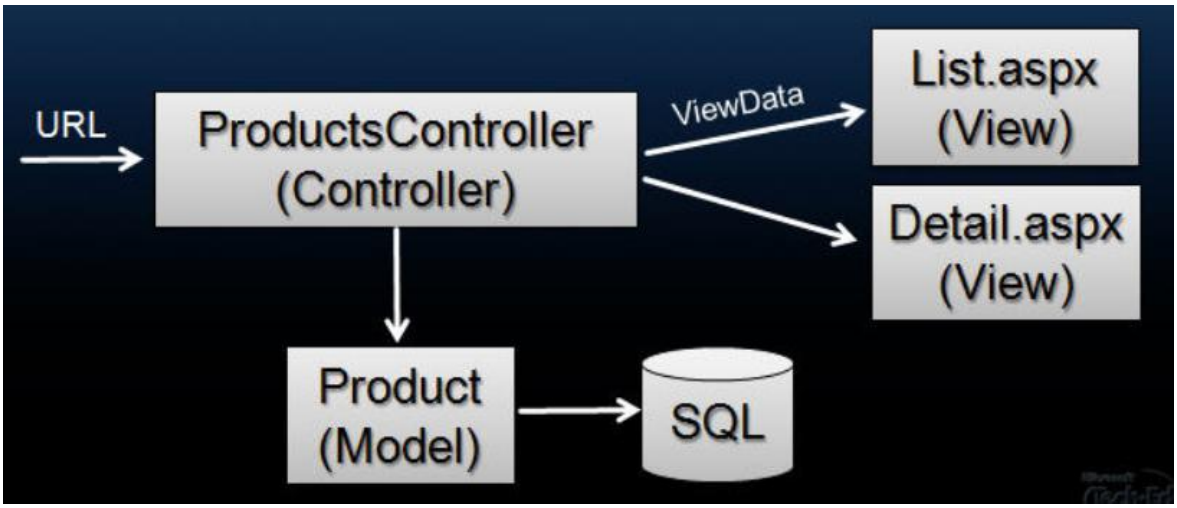
Controller mặc định sẽ là HomeController, action mặc định là Index, Id mặc định là " "

```
controller = "Home", action = "Index", id = UrlParameter.Optiona
```

Ánh xạ các URL vào trong class Controller

Hầu hết các Web Framework, như ASP , PHP , JSP , ASP.NET WebForms …, đều ánh xạ các URL vào một file được lưu trên đĩa. Lấy ví dụ URL "/Giaotrinh.aspx" hay "/Giaotrinh.php" được chuyển đến file Giaotrinh.aspx hay Giaotrinh.php trên đĩa cứng để xử lý. Khi một ứng dụng web nhận được HTTP Request đến Web Server, thì Web Framework sẽ chạy một đoạn code cụ thể tương ứng với nội dung của file, và đoạn code này đóng vai trò xử lý yêu cầu do phía client gửi đến. Thông thường thì đoạn code này sẽ sinh ra HTML và đáp ứng lại phía client.

MVC Framework lại hoạt động theo một cách hoàn toàn khác, thay vì ánh xạ các URL vào các file lưu trên đĩa, nó sẽ đưa thẳng vào các lớp (class). Những lớp (class) được ánh xạ tới được gọi là "Controllers", và chúng sẽ xử lý yêu cầu (request) được yêu cầu đến, kiểm soát dòng nhập xuất và giao diện đối với người dùng, thực thi các ứng dụng và data logic tương ứng với yêu cầu (request). Cuối cùng, chúng sử dụng các thành phần Views để tạo HTML và đáp trả lại yêu cầu (request).



Hình 3.5: Mô hình hoạt động của MVC

3.2.1. Hệ thống định tuyến trong ASP.NET MVC để làm gì ?

ASP.NET MVC Framwork có một hệ thống định tuyến URL (URL Routing System) linh hoạt cho phép xác định các quy tắc ánh xạ địa chỉ URL bên trong ứng dụng. Một hệ thống định tuyến có 2 mục đích:

- Xây dựng một tập hợp các URL đi vào ứng dụng và định tuyến chúng tới các Controller và thực thi các phương thức Action để xử lý.
- Xây dựng các URL gởi đi mà có thể gọi ngược trở lại Controller/Action.

Sử dụng các quy tắc ánh xạ URL để điều khiển URL đi vào và đi ra để tăng tính mềm dẻo cho việc lập trình ứng dụng, nghĩa là nếu muốn thay đổi cấu trúc URL (ví dụ /Giao_trinh bằng /Lop) có thể thay đổi một tập hợp quy tắc ánh xạ mức ứng dụng mà không cần phải viết lại mã lập trình bên trong Controllers và View.

3.2.2. Các quy tắc định tuyến các URL mặc định trong ASP.NET MVC Web Application

Mặc định khi tạo ứng dụng với ASP.NET MVC Web Application trong Visual Studio sẽ tạo ra một ASP.NET MVC Application Class gọi là Global.asax chứa cấu hình các quy tắc định tuyến URL. Xây dựng các định tuyến thông qua phương thức RegisterRoutes(ReouteCollection router) và khi ứng dụng bắt đầu, ứng dụng Application_Start() trong Global.asax sẽ gọi RegisterRouter để tạo bảng định tuyến.

```
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Web.Routing;

namespace mvcdemo
{
    // Note: For instructions on enabling IIS6 or IIS7 classic mode,
    // visit http://go.microsoft.com/?LinkId=9394801

    public class MvcApplication : System.Web.HttpApplication
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                "Default", // Route name
                "{controller}/{action}/{id}", // URL with parameters
                new { controller = "Home", action = "Index", id = UrlParameter.Optional } // Parameter defaults
            );

        }

        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();

            RegisterRoutes(RouteTable.Routes);
        }
    }
```

Nguyên tắc ánh xạ đầu tiên trong hình trên chỉ ra cho ASP.NET MVC Framework một quy luật ánh xạ URL theo dạng /[controller]/[action]/[id] – tương ứng với tên controller được ánh xạ đến / tên action method được triệu gọi / và tham số để truyền vào action method đó.

Với ASP.NET MVC Web Application thì mặc định controllers là HomeController, mặc định ControllerAction là Index và Id là rỗng

Khi ứng dụng ASP.NET MVC Web Application nhận được một URL, MVC FrameWork sẽ định giá các quy tắc định tuyến trong tập hợp RouteTable.Routes để quyết định Controller nào sẽ điều khiển request.

MVC Framwork sẽ chọn controller bằng cách định giá các quy tắc trong bảng định tuyến theo trật tự sẵn có.

Route Instances

Quy tắc ánh xạ được đăng ký bằng cách khai báo thêm một Route instances vào System.Web.Mvc.RouteTable's Route colletion.

Class Route này có một số thuộc tính mà bạn có thể sử dụng để cấu hình (configure) quy tắc ánh xạ của bạn. Bạn có thể "set" những thuộc tính đó theo cách truyền thống của .NetFramework 2.0

```
Route myRoute = new Route();

myRoute.Url = "[controller]/[action]/[id]";
myRoute.RouteHandler = typeof(MvcRouteHandler);
myRoute.Defaults = new { action = "Index", id = (string)null };

RouteTable.Routes.Add(myRoute);
```

Hoặc bạn có thể tận dụng tính năng object initializer của .NetFramework 3.5 trở lên

```
RouteTable.Routes.Add(new Route
{
    Url = "[controller]/[action]/[id]",
    Defaults = new { action = "Index", id = (string)null },
    RouteHandler = typeof(MvcRouteHandler)
});
```

Như hình trên ta có thể thấy các thuộc tính trong lớp Route cụ thể như sau:

Thuộc tính Url trong lớp Route dùng để khai báo một Pattern URL – một quy tắc ánh xạ URL để khi một URL được yêu cầu đến web của chúng ta, MVC Framework sẽ tự động ánh xạ URL đó đến Pattern này và phân tích các thành phần trong URL đó để biết đâu là Controller , đâu là Action Method và đâu là tham số đầu vào cho Action Method đó. Bạn không bị giới hạn bởi một tham số duy nhất mà chúng ta có thể có một bất kỳ số lượng tham số nào mà bạn muốn có trong URL.

Ví dụ bạn có thể sử dụng quy tắc ánh xạ "/Blogs/[Username]/Archive/[Year]/[Month]/[Day]/[Title]" để mã hóa một URL yêu cầu đến và MVC Framework sẽ tự động phân tích và truyền những tham số như là Username, Year, Month, Day và Title đến Action Method trong Controller của chúng ta.

Thuộc tính Default trong class Route dùng để khai báo một tập giá trị mặc định được sử dụng để xử lý các URL được yêu cầu đến không có các giá trị tham số như đã định ở thuộc tính Url. Ví dụ quy tắc ánh xạ URL bên trên chúng ta khai báo

2 tham số mặc định trong Url là Action và Id. Điều này có nghĩa là nếu một URL: "/ Quanly /" được yêu cầu đến thì hệ thống ánh xạ sẽ mặc định sử dụng "Index" như là một tên Action Method trong QuanlyController, cụ thể là Action Method Index() trong QuanlyController sẽ được thực thi xử lý khi người dùng yêu cầu Url "/ Quanly/". Tương tự, nếu như Url "/Quanly/Danhsachgt" được yêu cầu thì một giá trị tham số null sẽ được sử dụng cho Action Method Danhsachgt().

Thuộc tính RouteHandler trong class Route khai báo một IRouteHandler cụ thể được sử dụng để xử lý yêu cầu sau khi URL được mã hóa và xác định được quy tắc ánh xạ thích hợp. Trong ví dụ trên chúng ta chỉ cho MVC Framework rằng chúng ta muốn sử dụng class System.Web.Mvc.MvcRounteHandlerđể xử lý các URL mà chúng ta đã cấu hình. Lý do cho việc này là chúng ta muốn chắc rằng hệ thống ánh xạ URL được sử dụng cho cả hai trường hợp được người dùng yêu cầu là MVC và non-MVC (WebForms).

Ngoài ra còn một thuộc tính nữa trong class Route mà chúng ta sẽ tìm hiểu sau torng bài viết này. Nó cho phép chúng ta xác định trước những điều kiện cần thiết để áp dụng cho một quy tắc ánh xạ cụ thể. Ví dụ chúng ta có thể chỉ muốn quy tắc ánh xạ chỉ áp dụng cho HTTP cụ thể, hoặc chúng ta có thể dùng Regular Expression như những tham số để lọc những quy tắc ánh xạ phù hợp…

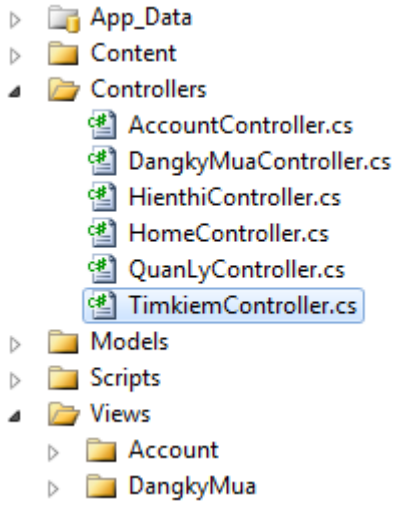
Route Rules Evaluation

Khi một URL được yêu cầu đến ASP.NET MVC Application, MVC Framework tìm trong bảng hệ thống ánh xạ (RouteTable.Routes collection) để xác định một quy tắc ánh xạ thích hợp để xử lý. MVC Framework xác định Controller bằng cách tìm trong những quy tắc ánh xạ mà chúng ta đã tạo theo tứ tự từ trên xuống dưới. URL được yêu cầu đến sẽ được so sánh với từng quy tắc ánh xạ trong RouteTable.Routes collection, nếu một quy tắc ánh xạ nào phù hợp với URL đó thì nó sẽ được áp dụng (tất cả những quy tắc còn lại sẽ được bỏ qua). Điều này có nghĩa là bạn phải sắp xếp các quy tắc ánh xạ một cách thật hợp lý.

Routing Scenario: Custom Search URL

Hãy sử dụng các quy tắc ánh xạ để làm một kịch bản thực tế bằng cách chúng ta sẽ thực hiện chức năng tìm kiếm cho trang web đăng ký mua giáo trình của chúng ta.

Đầu tiên ta sẽ thêm một lớp *TimkiemController* cho ứng dụng:



Hình 3.6: Giao diện thêm lớp TimkiemController

Ta sẽ định nghĩa hai Action Method trong class TimkiemController. Action Method Index() sẽ được sử dụng để trình bày một trang tìm kiếm, trong trang tìm kiếm đó sẽ có có một TextBox cho phép người dùng nhập vào và gửi một từ khóa tìm kiếm. Các Action Method Result() sẽ được sử dụng để xử lý khi người dùng Submit và thực hiện việc tìm kiếm cơ sở dữ liệu, và sau đó hiển thị kết quả lại cho người dùng:

TimkiemController.cs | Danhsachgt.aspx | QuanLyController.cs* | Global.asax.cs | Index.ascx | Muasach.aspx

Totnghiep.Controllers.TimkiemController | Timkiem()

```
using System.Web;
using System.Web.Mvc;
using Totnghiep.Models;

namespace Totnghiep.Controllers
{
    public class TimkiemController : Controller
    {
        //
        // GET: /Timkiem/
        csdlDataContext db = new csdlDataContext();

        public ActionResult Index(string tenGt,int? tien)
        {

            return View(from a in db.Giao_trinhs.Where(x => x.Ten_giao_trinh.Equals(tenGt) && x.Gia_tien == tien) select a);
        }
```

Sử dụng quy tắc ánh xạ mặc định /[controller]/[action]/[id] , Chúng ta sẽ sử dụng những URL sau đây để gọi xử lý cho chức năng tìm kiếm sản phẩm thông qua SearchController:

Bảng 3.1: Các Action Method theo URL

URL	Action Method
/Timkiem/?tenGT=XML&tien=20001	Index

Lưu ý rằng khi ta sử dụng URL "/Timkiem/" là chúng ta sử dụng quy tắc định tuyến mặc định do Visual Studio tạo sẵn theo mặc định khi chúng ta tạo ứng dụng này và như hồi nãy ta đã đề cập về thuộc tính Default trong class Route, MVC Framework sẽ sử dụng thuộc tính Default này để gọi Action Method Index() trong TimkiemController để xử lý URL này.

```
routes.MapRoute(
    "Default", // Route name
    "{controller}/{action}/{id}", // URL with parameters
    new { controller = "Home", action = "Index", id = UrlParameter.Optional } // Parameter defaults
);
```

Đối với URL "/Timkiem/?tenGT=XML&tien=20001". Nếu bạn muốn URL của chúng ta được rõ ràng hơn, minh bạch hơn thì chúng ta truyền tham số thông qua một QueryString, và coi đó như là một tham số trong phần thông số của URL.

Chúng ta có thể làm cho những URL tìm kiếm được đẹp hơn bằng cách thêm quy tắc ánh xạ URL trước quy tắc mặc định "/[controller]/[action]/[id]" mà Visual Studio đã tạo cho chúng ta khi tạo ứng dụng này như dưới đây:

```
routes.MapRoute(
    "Timkiem", // Route name
    "{Timkiem}/{action}/{id}", // URL with parameters
    new { controller = "Timkiem", action = "Index", id = UrlParameter.Optional } // Parameter defaults
```

Với quy tắc trên ta đã xác định rõ ràng các thông số Controller và những tham số cho Action Method cho URL " /Timkiem/". Chúng ta đang chỉ ra rằng URL "/Timkiem" luôn luôn phải được xử lý bởi Action Method Index() trong TimkiemController.

Validation Pre-Conditions for Routing Rules – Kiểm tra các điều kiện cần thiết cho một quy tắc ánh xạ

Như ta đã đề cập trước đó trong đồ án này, lớp Route có một thuộc tính "Validation" cho phép chúng ta thêm các điều kiện xác nhận trước khi quy tắc đó được MVC Framework đánh giá là phù hợp (ngoài các bộ lọc URL) cho một URL được yêu cầu đến. ASP.NET MVC Framework cho phép ta sử dụng Regular Expression để kiểm tra mỗi đối số trong URL, cũng như cho phép ta đánh giá các tiêu đề HTTP.

Dưới đây là một quy tắc validation mà ta muốn ứng dụng đối với URL như " /Quanly/Editgiaotrinh/1" Nó quy định rằng các đối số ID phải là một số (không chấp nhận một string), và rằng nó phải có từ 1 đến 8 ký ký tự:

```
routes.MapRoute(
    "Default", // Route name
    "{controller}/{action}/{id}", // URL with parameters
    new { controller = "Home", action = "Index"},
   new {id = @"\d{1,8}"}// Parameter defaults
);
```

Nếu chúng ta có một URL được yêu cầu đến như /Quanly/Editgiaotrinh/1 " thì quy tắc ánh xạ trên sẽ có hiệu lực. Nếu chúng ta có một URL được yêu cầu đến như

/Quanly/Editgiaotrinh/tri hoặc /Quanly/Editgiaotrinh/1212121212 thì nó sẽ không phù hợp.

Constructing Outgoing URLs from the Routing System – Xây dựng các Outgoing URL từ hệ thống các quy tắc ánh xạ

Hệ thống ánh xạ URL trong ASP.NET MVC Framework chịu trách nhiệm về hai điều:

- Ánh xạ các URL được yêu cầu đến các Action Method trong các class Controller để xử lý.
- Giúp đỡ xây dựng ra các URL có thể được sử dụng để gọi về Action Method trong lớp Controller.

Hệ thống ánh xạ URL có một số phương thức bổ trợ (Helper Method) và các lớp để cho nó dễ dàng tự động tìm kiếm và xây dựng các URL trong thời gian chạy (runtime).

Html.ActionLink

Helper Method Html.ActionLink. Nó có thể được sử dụng trong các thành phần View và cho phép bạn tự động tạo ra siêu liên kết <a href=””"> dựa trên các quy tắc ánh xạ URL của chúng ta được khai báo trong hệ thống ánh xạ trong tập tin Global.asax. Ví dụ:

```
<tr>
    <td>
        <%: Html.ActionLink("Chỉnh sửa", "Editgiaotrinh", new { id=item.Ma_giao_trinh }) %> |
        <%: Html.ActionLink("Xóa", "Deletegt", new { id=item.Ma_giao_trinh })%>
    </td>
```

Nó tự động chọn những quy tắc ánh xạ phù hợp trong hệt thống ánh xạ mà ta đã định nghĩa trước đó trong ví dụ này và sẽ tự động tạo ra giá trị “href” cho các siêu liên kết.

Url.Action

Ngoài việc sử dụng Html.ActionLink, ASP.NET MVC còn có một *Helper Method* **Url.Action()** . *Helper Method* này tạo ra chuỗi URL thô – mà bạn có thể sử dụng nếu bạn muốn. Nó sẽ sử dụng hệ thống ánh xạ URL để gen ra một chuỗi như dưới đây (không có trong thẻ a).

Controller.RedirectToAction

ASP.NET MVC cũng hỗ trợ một Helper Method Controller.RedirectToAction() mà ta có thể sử dụng trong điều khiển để thực hiện việc chuyển hướng (Redirect) (trong đó các URL được xác định bằng cách sử dụng hệ thống ánh xạ URL).

Ví dụ khi mã dưới đây được gọi trong một Controller :

```
return RedirectToAction("Quanly/danhsachgt");
```

thì nó sẽ tương ứng với Response.Redirect(“/Quanly/danhsachgt”) (điều này đã quá quen thuộc với chúng ta khi lập trình với ASP.NET Web Form).

Điểm hay ở những Helper Medthod trên là nó cho phép chúng ta tránh các hard-code trong URL bên trong Controller và . Nếu sau này bạn muốn thay đổi quy tắc ánh xạ URL từ “/Search/[query]/[page]“ sang “/Search/Results/[query]/[page]” hoặc “/Search/Results?query= [query]&page=[page]“. Chúng ta chỉ cần thay đổi mã nguồn ở một nơi duy nhất mà không cần phải thay đổi thêm bất cứ mã nguồn nào trong các thành phần Controller và View của chúng ta.

Constructing Outgoing URLs from the Routing System (using Lambda Expressions) - Xây dựng các URL gửi đi từ hệ thống ánh xạ (sử dụng biểu thức Lambda)

Ngoài việc truyền thông số một cách linh hoạt bằng cách sử dụng anonymous type, ASP.NET MVC Framework cũng hỗ trợ khả năng tạo ra các ánh xạ bằng cách sử dụng strongly-typed một cơ chế cho cho phép compile-time checking và intellisense cho URL Helper. Đó là các biểu thức Lambda.

Ví dụ, ActionLink anonymous type:

```
<%= Html.ActionLink("Search Drinks", new { Controller = "Search", Action = "Results", Query="Beverages", Page
```

Có thể được viết lại như sau:

```
<%= Html.ActionLink<SearchController>( "Search Drinks", s=>s.Results("Beverages", 2)) %>
```

Chú ý: Khi sử dụng Lambda Expression chúng ta không bao giờ thực sự thực thi các Action Method trong SearchController. Ví dụ, mã dưới đây không gọi Action Method "Result":

```
<%= Html.ActionLink<SearchController>( "Search Drinks", s=>s.Results("Beverages", 2)) %>
```

mà chỉ tạo ra mã HTML của một siêu liên kết:

```
<a href="/Search/Beverages/2">Search Drinks</a>
```

Khi người dùng nhấp vào liên kết này nó sẽ gửi lại một yêu cầu http đến máy chủ và sẽ gọi Action Method Result của SearchController.

3.3. Xây dựng Controllers

Mỗi lần có một yêu cầu được gởi đến trang web ASP.MET MVC của bạn thì nó sẽ được giải quyết bởi controller. Controllers có trách nhiệm điều khiển các tương tác của người dùng bên trong ứng dụng MVC. Controllers quyết định cái gì sẽ được chuyển về cho người dùng khi tạo một yêu cầu (request) trên trình duyệt (browser).

Controller có trách nhiệm cho các ứng dụng logic, bao gồm tiếp nhận thông tin người dùng nhập vào, ra lệnh, lấy dữ liệu từ Model và cuối cùng là xử lý. Có một số tương đồng giữa các Controller ASP.NET MVC và các trang ASPX trong ASP.NET Webform. Ví dụ, cả hai đều là các điểm tương tác với người dùng cuối. Tuy nhiên trong một số cách khác, nó có các khái niệm khá khác nhau, ví dụ:

- Ta không thể tách rời trang ASPX với code xử lý logic của nó, vì cả hai nó đều hợp tác để thực hiện các ứng dụng nhất định. Tuy nhiên ASP.NET MVC thì lại khác , chúng tách biệt thành phần giao diện với phần code xử lý, chính vì thế ta có thể giữ cho code của mình được đơn giản, dễ hiểu, duy trì sự cô lập cần thiết.
- Trang ASP.NET WebForm ASP có một liên kết một - một với một giao diện người dùng cụ thể. Tuy nhiên trong ASP.NET MVC, một bộ điều khiển không gắn liền với một giao diện người dùng cụ thể, vì vậy nó có thể giải quyết các yêu cầu bằng cách gởi lại các dữ liệu cần thiết.

Tất nhiên là những ứng dụng thực tế sẽ giúp chúng ta hiểu rõ về Controller hơn, xem nó có thể giúp chúng ta xây dựng những ứng dụng một cách đơn giản như thế nào.

Các lớp Controller cơ bản

MVC Framwork đi kèm một lớp cơ sở tiêu chuẩn để điều khiển, System.Web.Mvc.Controller. Bao gồm các thành phần sau đây:

- Action methods: hành động của chúng ta được chia thành nhiều phương thức, mỗi phương thức tương ứng với một địa chỉ URL khác nhau, và được gọi với các tham số được lấy từ yêu cầu.
- Action results: ta có thể tùy chọn để trả về một đối tượng mô tả những dự định kết quả của một hành động (ví dụ có thể trả về một View, đến một Action method khác…) và nó sẽ thực hiện cho chúng ta. Việc tách bạch giữa xác định kết quả với việc thực thi sẽ đơn giản hóa việc kiểm thử một cách đáng kể.
- Filters: ta có thể rút gọn các hành vi sử dụng lại được (ví dụ như chứng thực) như bộ lọc, và sau đó khóa mỗi hành vi vào một hoặc nhiều controller hay action moethod bằng cách đặt một [thuộc tính] trong mã nguồn của chúng ta.

Một controllers là một lớp (Class) (C# class hoặc VB class). Trong ví dụ ứng dụng ASP.NET MVC Web Application mẫu luôn tồn tại 2 controllers là AccountController.cs và HomeController.cs nằm trong folder Controllers.

HomeController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace Hitest.Controllers
{
   [HandleError]
   public class HomeController : Controller
   {
      public ActionResult Index()
      {
         ViewData["Message"] = "Welcome to ASP.NET MVC!";
         return View();
      }
      public ActionResult About()
      {
         return View();
      }
   }
}
```

Trong HomeController.cs có 2 phương thức là Index() và About(). Cả 2 phương thức này là 2 action trong controller HomeController.cs, nó thực hiện khi được gọi bằng địa chỉ /Home/Index và Home/About. Bất cứ phương thức nào có thuộc tính public đều là một action trong controller.

Tìm hiểu về các Action Method trong Controler.

Sau khi ta tạo một lớp QuanlyController, ta có thể bắt đầu thêm các businness logic để xử lý việc khi người dùng truy cập vào URL "/ Quanly/ Danhsachgt" của ứng dụng.

Trong ví dụ này chúng ta sẽ thực hiện 3 kịch bản cho ứng dụng Web của chúng ta :

- Duyệt tất cả các danh sách giáo trình
- Chỉnh sửa thông tin giáo trình
- Thêm giáo trình mới

Chúng ta sẽ sử dụng các URL sau đây để xử lý từng tình huống:

Bảng 3.2: Xử lý các URL

Định dạng URL	Hành vi	Ví dụ URL
/Quanly/Danhsachgt	Duyệt tất cả các danh sách giáo trình	*/Quanly/Danhsachgt*
/Quanly/Editgiaotrinh/Magt	Chỉnh sửa thông tin giáo trình	*/Quanly/Editgiaotrinh/1*
/Quanly/Creategt	Thêm mới giáo trình	*/Quanly/Creategt*

Có một vài cách để chúng ta viết lệnh (code) trong lớp *QuanlyController* để xử lý ba loại URL được gọi đến.

Một cách là ta có thể nạp chồng (Override) lại phương thức "Execute" trên lớp Base Controller và viết các câu lệnh if / else / switch logic bằng tay để xem các URL được yêu cầu đến và sau đó thực hiện các logic thích hợp để xử lý nó.

Một cách dễ dàng hơn nhiều, là sử dụng tính năng được tích hợp của MVC Framework cho phép chúng ta định nghĩa "Action Method" trên Controller, và sau đó lớp Base Controller tự động gọi Action Method thích hợp để thực hiện dựa trên quy tắc ánh xạ URL sử dụng cho ứng dụng.

Ví dụ, chúng ta sẽ thêm 3 Action Method vào class QuanlyController để xử lý các URL được yêu cầu đến như sau:

```
public ActionResult Danhsachgt()
{
    var gt = csdl.Giao_trinhs;
    return View(gt.ToList());
}

[HttpGet]
public ActionResult Creategt()
{
    return View();
}

[HttpPost]
public ActionResult Creategt([Bind(Exclude = "Ma_giao_trinh")]Giao_trinh gt)...
[HttpGet]
public ActionResult Editgiaotrinh(int id)
{
    var editgt = csdl.Giao_trinhs.Select(p => p).Where(p => p.Ma_giao_trinh == id).FirstOrDefault();
    return View(editgt);
}

[HttpPost]
public ActionResult Editgiaotrinh(Giao_trinh gt)...
```

Các quy tắc của việc ánh xạ URL đã được cấu hình mặc định khi một dự án mới được tạo. Vì vậy, nếu chúng ta nhận được một yêu cầu URL /Quanly/Danhsachgt, các quy tắc ánh xạ sẽ xử lý "Danhsachgt" là tên của một ActionMethod trong QuanlyController, và phương thức Danhsachgt () này sẽ được gọi để xử lý yêu cầu. Tương tự nếu chúng ta nhận được một yêu cầu URL /Quanly/Editgiaotrinh/1, quy tắc định tuyến sẽ xử lý "Editgiaotrinh" như tên của một ActionMethod, và phương thức Editgiaotrinh (int id) sẽ được gọi để xử lý yêu cầu, v…v…

Ánh xạ URL có tham số đến Action Methods trong Controller

Có nhiều cách để lấy được tham số trong URL từ những Action Method của các lớp Controller .

Các lớp Controller của chúng ta được implement (thực thi các phương thức từ một lớp) từ một Base Controller – nó đưa ra một tập hợp các đối tượng Request (yêu cầu) và Reponse (đáp ứng) có thể được sử dụng. Những đối tượng này có chính xác cùng một cấu trúc API (Application Programming Interface - giao diện lập trình ứng dụng) như là HttpRequest / HttpResponse – đối tượng mà bạn đã quen

thuộc với trong ASP.NET Web Form . Điều khác biệt quan trọng là các đối tượng này implement (thực thi các phương thức từ một lớp) từ các giao diện (interface) thay vì các lớp được niêm phong – không được thừa kế (sealed classes). Lợi ích của việc có các giao diện (interface) này là ta dễ dàng móc nối với chúng – cho phép dễ dàng sử dụng Unit Testvới các lớp Controller.

Dưới đây là một ví dụ về cách chúng ta có thể sử dụng Request API để lấy một giá trị chuỗi truy vấn "ID" từ bên trong Action Method Editgiaotrinh() của chúng ta trong lớp QuanlyController:

```
public ActionResult Editgiaotrinh(int id)
{
    var editgt = csdl.Giao_trinhs.Select(p => p).Where(p => p.Ma_giao_trinh == id).FirstOrDefault();
    return View(editgt);
}
```

ASP.NET MVC Framework cũng hỗ trợ tự động ánh xạ các giá trị tham số từ URL đến Action Method như như một tham số của Action Method đó. Theo mặc định nếu bạn có một tham số trong Action Method của bạn, MVC Framework sẽ xem xét các dữ liệu yêu cầu gửi đến để xem có một yêu cầu HTTP tương ứng có giá trị cùng tên nào không. Nếu có, nó sẽ tự động chuyển nó vào như một tham số cho Action Method của bạn.

Ngoài ra ASP.NET MVC Framework cũng cho phép bạn sử dụng URL Routing để truyền tham số (ví dụ: thay vì Quanly/Editgiaotrinh?id=34 bạn có thể sử dụng Quanly/Editgiaotrinh/34).

Quy luật ánh xạ tên là "Default" đã được khai báo mặc định trong file Global.asax từ khi chúng ta tạo Project này có định dạng "/[controller]/[action]/[id]". Điều này có nghĩa là nếu có bất kỳ đường dẫn URL được yêu cầu đến thì hậu tố sau cùng của nó (sau tên của Controller và tên Action Method) thì theo mặc định sẽ được coi như một tham số có tên "id" – và có thể được tự động truyền vào Action Method của chúng ta như là một tham số đầu vào.

Từ đó Action Method Detail() cũng sẽ nhận được tham số "ID" từ đường dẫn URL (ví dụ: /Quanly/Editgiaotrinh/1):

Dưới đây là một lớp QuanlyController của em bây giờ đã có các Action Method cụ thể xử lý từng URL được yêu cầu với các tham số cần thiết:

```
public ActionResult Danhsachgt()
      {
      }

      [HttpGet]
      public ActionResult Creategt()
      {
         return View();
      }

      [HttpPost]
      public ActionResult Creategt([Bind(Exclude = "Ma_giao_trinh")]Giao_trinh
gt)
      {
      }
      [HttpGet]

      public ActionResult Editgiaotrinh(int id)
      {
      }

      [HttpPost]
      public ActionResult Editgiaotrinh(Giao_trinh gt)
      {
      }
```

Tham số tùy chọn trong MVC Framework được sử dụng theo kiểu dữ liệu nullable trong Action Method của chúng ta. Vì tham số chỉ mục trên Action Method Editgiaotrinh() là một biến int (có cú pháp khai báo là "int?"), nên MVC Framework sẽ bỏ qua giá trị này nếu nó không có mặt trong URL – và sẽ truyền vào Action Method của ta nếu nó có mặt trong URL.

Các loại Action Result

Bảng 3.3: Các loại Action Result

Các kiểu trả về	Mục đích	Ví dụ về sử dụng
ViewResult	Hiển thị một View mới hoặc trang mặc định.	Return View(); Return View("MyView", modelObject);
PartialViewResul	Trả về một View mới hoặc	Return PartialView();

t	mặc định cục bộ.	Return PartialView("MyPartial", modelObject);
RedirectToRoute Result	Trả về một Action method mới	ReturnRedirectToAction("SomeOtherAction", "SomeController"); ReturnRedirectToRoute("MyNamedRoute");
RedirectResult	Chuyển tới một địa chỉ khác	ReturnRedirect("http://www.example.com");
ContentResult	Trả về dữ liệu thô văn bản đến trình duyệt	ReturnContent(rssString, "application/rss+xml");
FileResult	Truyền dữ liệu nhị phân (chẳng hạn như tập tin từ ổ đĩa) trực tiếp đến trình duyệt.	Return File(@"c:\report.pdf", "application/pdf");
JsonResult	Chuyển đổi một đối tượng sang kiểu Json và trả về như một phản hồi.	ReturnJson(someObject);
JavaScriptResult	Gửi một đoạn mã nguồn JavaScript cần phải được thực hiện bởi trình duyệt. Đây chỉ dùng để sử dụng trong các kịch bản Ajax	ReturnJavaScript("$('#myelem').hide();");
HttpUnauthorized Result	Thiết lập trạng thái phản ứng HTTP mã 401 (có nghĩa là "không được uỷ quyền"), mà nguyên nhân cơ chế xác thực hoạt động (Hình thức xác thực hoặc Windows Authentication) là yêu cầu người truy cập phải đăng nhập.	Return new HttpUnauthorizedResult();
EmptyResult	Không trả về cái gì.	Return new EmptyResult();

3.4. Xây dựng Model

Hiện tại chúng ta đã có một lớp QuanlyController và ba Action Method sẵn sàng để xử lý yêu cầu trang web gửi đến. Bước tiếp theo sẽ là xây dựng một số lớp để giúp chúng ta làm việc với cơ sở dữ liệu, lấy các dữ liệu thích hợp cần thiết để xử lý các yêu cầu web.

Trong một ứng dụng ASP.NET MVC, " Model " là các thành phần có trách nhiệm duy trì trạng thái của các đối tượng, thông thường nó là một lớp ánh xạ đến một bảng trong cơ sở dữ liệu (ví dụ: chúng ta có một lớp Giao_trinh được sử dụng để mô tả bảng Giao_trinh bên trong cơ sở dữ liệu SQL của chúng ta).

ASP.NET MVC Framework cho phép bạn sử dụng bất kỳ mô hình truy cập dữ liệu nào bạn muốn để thao tác, quản lý dữ liệu của bạn. Bạn có thể sử dụng ADO.NET DataSets / DataReaders, hoặc nếu bạn thích sử dụng một mô hình ánh xạ đối tượng quan hệ (ORM) như NHibernate, LLBLGen, WilsonORMapper, LINQ to SQL / LINQ To Entities.

Đối với ứng dụng mua sách này chúng ta sẽ sử dụng LINQ to SQL class.

Ta sẽ bắt đầu bằng cách phải chuột trên thư mục, " Model " trong dự án web MVC của chúng ta trong VS và chọn "Add New Item" để thêm một mô hình LINQ to SQL .

Ta tạo một lớp Giao_trinh.cs.Trong lớp này ta sẽ khai báo các thuộc tính của bảng Giao_trinh, mục đích là sẽ tạo ra một lớp trung gian giữa cơ sở dữ liệu và Controller:

```
Totnghiep.Models.giaotrinhMetadata
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel;
using System.Web.Mvc;

namespace Totnghiep.Models
{
    [MetadataType(typeof(giaotrinhMetadata))]
    public partial class Giao_trinh
    {
    }
    public partial class giaotrinhMetadata
    {
        [DisplayName("Tên giáo trình")]
        [Required(ErrorMessage = "Yêu cầu nhập tên giáo trình")]
        public string Ten_giao_trinh { get; set; }
        [DisplayName("Giá tiền")]
        [Required(ErrorMessage = "yêu cầu nhập giá tiền")]
        public string Gia_tien { get; set; }
    }
}
```

Như vậy là chúng ta đã có tất cả dữ liệu mà ta cần cho lớp QuanlyController. Tiếp theo chúng ta hãy tiến hành cài đặt lớp QuanlyController.

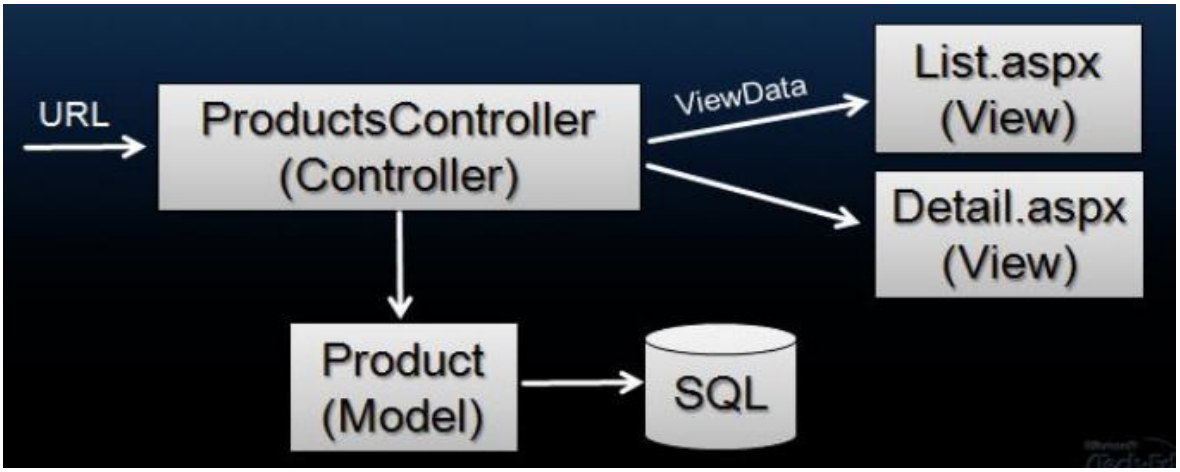
Cài đặt lớp QuanlyController

Controller trong một ứng dụng MVC có trách nhiệm xử lý các yêu cầu gửi đến, xử lý và tương tác của người dùng, và thực hiện logic thích hợp (truy xuất và cập nhật dữ liệu được lưu trữ trong một cơ sở dữ liệu, v…v…)

Controller trong một ứng dụng MVC không trả về một HTML cụ thể như ở ASP.NET Web Form mà nhiệm vụ tạo ra HTML được thực hiện bởi các thành phần " View " trong ứng dụng một cách riêng biệt. View chỉ lo việc trình bày dữ liệu, và không chứa bất kỳ buiness logic hoặc code truy cập cơ sở dữ liệu nào (mà thay vào đó chúng được xử lý bởi các Controller).

Trong một quy trình làm việc web MVC , các Action Method trong Controller sẽ xử lý các yêu cầu web được yêu cầu đến, sử dụng các giá trị tham số đến để thực thi các buiness logic, truy xuất hoặc cập nhật các đối tượng mô hình dữ liệu từ cơ sở dữ liệu, và sau đó chọn một " View " để render về một giao diện người dùng thích

hợp và Reponse về cho trình duyệt. Controller sẽ truyền một đối tượng dữ liệu Strongly Typed đến " View " để cho nó có thể render một giao diện phú hợp:



Hình 3.6: Mô hình hoạt động của MVC

Ta có thể tự hỏi – lợi ích của việc tách Controller và View như thế này là những gì ? Tại sao không đặt chúng trong cùng một class ? Động lực chính trong phân vùng các ứng dụng như thế này là giúp cho việc Unit Testing được dễ dàng, làm cho mã nguồn ứng dụng của chúng ta trong sáng hơn, rõ ràng hơn từ đó sẽ làm cho ứng dụng của chúng ta dễ bảo trì hơn theo thời gian.

Khi cài đặt ba Action Method của class QuanlyController của chúng ta, em sẽ sử dụng các giá trị tham số URL đến để lấy các mô hình đối tượng thích hợp từ cơ sở dữ liệu, và sau đó chọn một thành phần " View " để vẽ lại một Reponse HTML thích hợp. Chúng ta sẽ sử dụng phương thức RenderView () trong lớp Base Controller để xác định xem chúng ta muốn sử dụng cũng như truyền vào một dữ liệu cụ thể mà chúng ta muốn xem.

Dưới đây là kết quả cuối cùng lớp QuanlyController của chúng ta:

```
public ActionResult Danhsachgt()
    {
       var gt = csdl.Giao_trinhs;
       return View(gt.ToList());
    }

    [HttpGet]
    public ActionResult Creategt()
    {
       return View();
```

```
    }

    [HttpPost]
    public ActionResult Creategt([Bind(Exclude = "Ma_giao_trinh")]Giao_trinh
gt)
    {
        if (ModelState.IsValid)
        {
            csdl.Giao_trinhs.InsertOnSubmit(gt);
            csdl.SubmitChanges();

        return RedirectToAction("danhsachgt");
        }
        return View(gt);
    }
    [HttpGet]

    public ActionResult Editgiaotrinh(int id)
    {
        var editgt = csdl.Giao_trinhs.Select(p => p).Where(p => p.Ma_giao_trinh
== id).FirstOrDefault();
        return View(editgt);
    }

    [HttpPost]
    public ActionResult Editgiaotrinh(Giao_trinh gt)
    {
        var gtrinh = csdl.Giao_trinhs.Select(p => p).Where(p => p.Ma_giao_trinh
== gt.Ma_giao_trinh).FirstOrDefault();
        gtrinh.Ma_giao_trinh = gt.Ma_giao_trinh;
        gtrinh.Ten_giao_trinh = gt.Ten_giao_trinh;
        gtrinh.Gia_tien = gt.Gia_tien;
        csdl.SubmitChanges();

        return RedirectToAction("Danhsachgt");
    }
```

3.5. Tạo giao diện người dùng với View

Chúng ta đã hoàn tất việc triển khai thực hiện và thử nghiệm các ứng dụng của ứng dụng mua bán giáo trình. Bây giờ chúng ta cần phải thực hiện các giao diện người dùng HTML cho nó.

Chúng ta sẽ làm điều này bằng cách cài đặt " View " để tạo ra một giao diện người dùng thích hợp khi gọi RenderView () :

```
#region Quan ly giao trinh

public ActionResult Danhsachgt()
{
    var gt = csdl.Giao_trinhs;
    return View(gt.ToList());
}
```

Trong đoạn mã ví dụ trên tham số "Danhsachgt" của RenderView() chỉ là tên của một View mà chúng ta muốn render, và tham số thứ hai là một đối tượng danh sách các danh mục sản phẩm cụ thể mà chúng ta muốn truyền vào đối tượng View sử dụng như là dữ liệu để tạo ra HTML thích hợp cho việc tao giao diện người dùng .

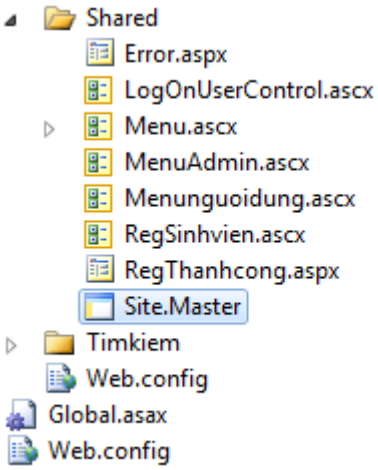
Các ASP.NET MVC Framework hỗ trợ khả năng sử dụng bất kỳ công cụ khuôn mẫu để giúp ta tạo ra các giao diện người dùng (bao gồm cả các công cụ template hiện có giống như NVelocity, Brail – cũng như những template mới mà bạn tự viết). Theo mặc định các ASP.NET MVC Framework hiện tại sử dụng trang ASP.NET (.aspx), Master Page (.master), và UserControl (.ascx) đã được hỗ trợ trong ASP.NET.

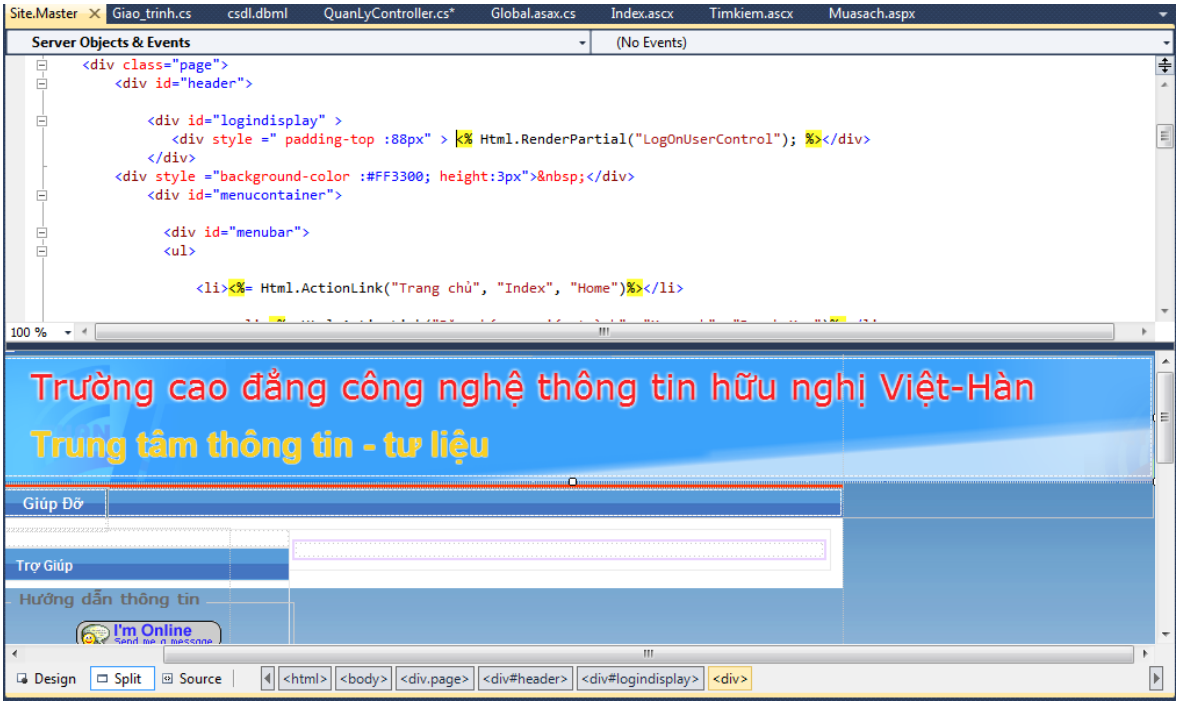
Chúng ta sẽ sử dụng công cụ xây dụng giao diện người dùng của ASP.NET để thực hiện cho ứng dụng mua giáo trình của chúng ta.

Định nghĩa một File Site.Master

Công dụng của trang **Master Page** chúng ta đã biết đến ở **ASP.NET Web Form** trước đây, đồ án này sẽ không đề cập đến.

Mặc định thì khi ta tạo Project MVC thì đã có sẵn một Site.Master trong folder Share.





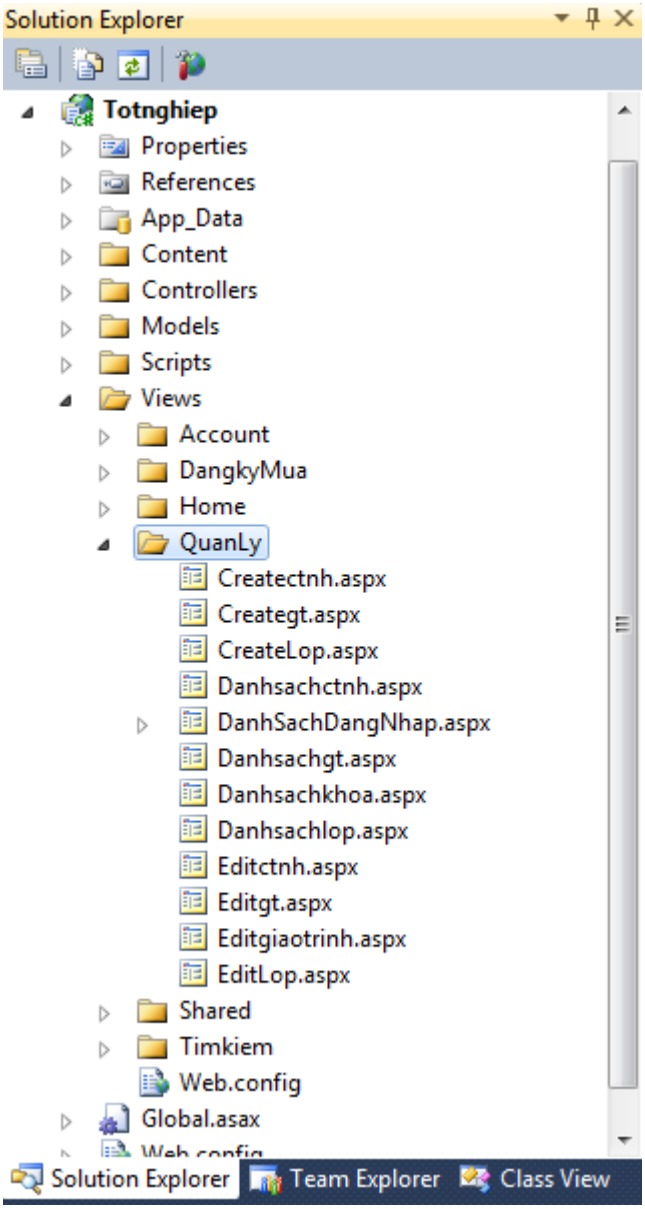
Hình 3.7: Giao diện trang Master.Page

Tìm hiểu cấu trúc thư mục View

Theo mặc định khi ta tạo ra một dự án ASP.NET MVC mới bằng cách sử dụng Visual Studio, nó sẽ tạo ra một thư mục "Shared" bên trong thư mục gốc "View" . Đây là nơi được đề nghị để lưu trữ Master Pages, UserControl , và Views mà ta muốn dùng nhiều nơi trong ứng dụng.

Khi xây dựng thành phần View cụ thể đối với một Controller, mặc định ASP.NET MVC sẽ lưu trữ chúng trong thư mục con trong thư mục gốc View. Theo mặc định, tên của một thư mục con phải tương ứng với tên điều khiển. Ví dụ , bởi vì

Controller chúng ta đã xây dựng được gọi là "QuanlyController", mặc định các View của QuanlyController sẽ được lưu trữ trong thư mục con tên là "Quanly":



Hình 3.9: View Quanly

Khi chúng ta gọi (invoke) câu lệnh RenderView(string viewName) trong một Controller , MVC Framework sẽ tự động tìm một View template tương ứng .Aspx hoặc .Ascx trong đường dẫn /View/ControllerName, nếu không thể tìm thấy View nào thích hợp thì ASP.NET MVC Framework sẽ tìm trong thư mục /View/Shared.

Tạo một Danhsachgt View

Chúng ta có thể tạo ra View "Danhsachgt" cho QuanlyController trong Visual Studio bằng cách click phải chuột chọn "Add New Item" trên thư mục Quanly và chọn "MVC View Page". Chúng ta sẽ có một trang aspx mới và có thể tùy chọn liên kết với trang chủ Site.Master.

Khi xây dựng ứng dụng bằng cách sử dụng một mô hình MVC , ta muốn giữ cho mã nguồn View đơn giản, và chắc chắn rằng mã nguồn View chỉ đơn thuần làm việc biểu diễn UI cho người dùng. Các xử lý về Buiness logic chỉ nên đặt trong các lớp Controller. Controller sẽ xử lý, chọn View thích hợp và truyền nhưng dữ liệu cần thiết vào View để biểu diễn ra UI cho người dùng (được thực hiện khi gọi RederView()). Ví dụ, dưới đây trong Action Method Danhsachgt của class QuanlyController, ta truyền vào Danhsachgt View một đối tượng "*var gt = csdl.Giao_trinhs*" để Categories View thực hiện việc hiển thị dữ liệu dựa vào đối tượng *gt*

```
#region Quan ly giao trinh

public ActionResult Danhsachgt()
{
    var gt = csdl.Giao_trinhs;
    return View(gt.ToList());
}
```

MVC View Page mặc định được kế thừa từ các lớp cơ sở System.Web.Mvc.ViewPage, cung cấp một số phương thức Helper và các Properties mà chúng ta có thể sử dụng trong việc xây dựng giao diện người dùng. Và đây là kết quả:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master"
Inherits="System.Web.Mvc.ViewPage<IEnumerable<Totnghiep.Models.Giao_trinh>>"
%>
<asp:Content ID="Content1" ContentPlaceHolderID="TitleContent" runat="server">
        Danh sách giáo trình
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
   <center>
```

```
<div class="top"></div>
<center ><div class="center"><doc>Danh sách giáo trình</doc></div> </center>
<div class="bottom"></div>
</center>
<hr />
<table width =722px>
    <tr>
        <th style="background-color:#2D8ECE; color :White ">Thao tác</th>
        <th style="background-color:#2D8ECE; color :White ">Tên giáo trình</th>
        <th style="background-color:#2D8ECE; color :White ">Giá tiền</th>
    </tr>
<% foreach (var item in Model) { %>
    <tr>
        <td>
            <%: Html.ActionLink("Chỉnh sửa", "Editgiaotrinh", new {
id=item.Ma_giao_trinh }) %> |
            <%: Html.ActionLink("Xóa", "Deletegt", new { id=item.Ma_giao_trinh })%>
        </td>
        <td>
            <%: item.Ten_giao_trinh %>
        </td>
        <td>
            <%: item.Gia_tien %>
        </td>
    </tr>

<% } %>

</table>
<p>
    <%: Html.ActionLink("Thêm mới", "Creategt") %>
</p>
</asp:Content>
```

Hình 3.10: Danh sách giáo trình

3.6. Truy nhập dữ liệu với LINQ

LINQ (Language Integrate Query) là sự sáng tạo mới trong .Net Framework 3.5, là một tập mở rộng ngôn ngữ cho phép thực hiện các truy vấn trong ngôn ngữ C# 2008 và VisualBasic 2008. LINQ cho phép Select (chọn), Filter (lọc), Sort (phân loại), Group (nhóm) và tranfom data (chuyển dữ liệu) từ các nguồn data source (dữ liệu nguồn) khác nhau theo một cách chung.

- LINQ to Objects thực hiện truy vấn các đối tượng.
- LINQ to DataSet thực hiện truy vấn DataSet.
- LINQ to SQL thực hiện truy vấn đến cơ sở dữ liệu SqlServer mà không phải viết code.
- LINQ to XML đọc dữ liệu từ XML.

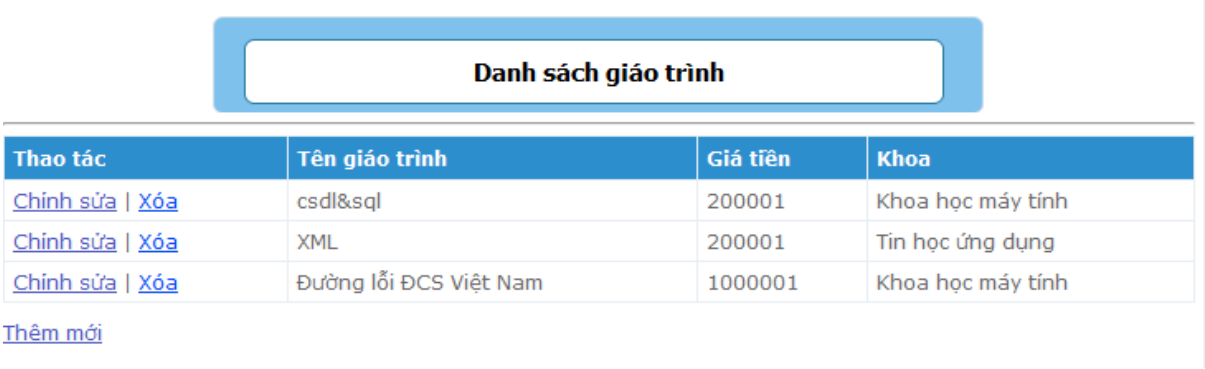
Ví dụ, nếu em không dùng LINQ, thì để muốn lấy tất cả các giáo trình thì em viết mã lệnh như sau:

```
public ActionResult Danhsachgiaotrinh()
{
    SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["csdlConnectionString"].ConnectionString);
    SqlCommand sql = new SqlCommand("select * from Giao_trinh", conn);
    SqlDataAdapter sda = new SqlDataAdapter(sql);
    DataTable bang = new DataTable();
    sda.Fill(bang);
   ViewData["giaotrinh"]= bang.DefaultView;
    return View();

}
```

Tuy nhiên nếu sử dụng LINQ thì mã lệnh như sau:

```
public ActionResult Danhsachgiaotrinh()
{
    var gt = csdl.Giao_trinhs;
    return View(gt.ToList());

}
```

Như vậy thì với việc sử dụng LINQ, câu lệnh truy vấn đã trở nên gọn gàng hơn rất nhiều, và đây là kết quả:



Danh sách giáo trình

Thao tác	Tên giáo trình	Giá tiền	Khoa
Chỉnh sửa | Xóa	csdl&sql	200001	Khoa học máy tính
Chỉnh sửa | Xóa	XML	200001	Tin học ứng dụng
Chỉnh sửa | Xóa	Đường lỗi ĐCS Việt Nam	1000001	Khoa học máy tính

Thêm mới

Hình 3.11: Danh sách giáo trình

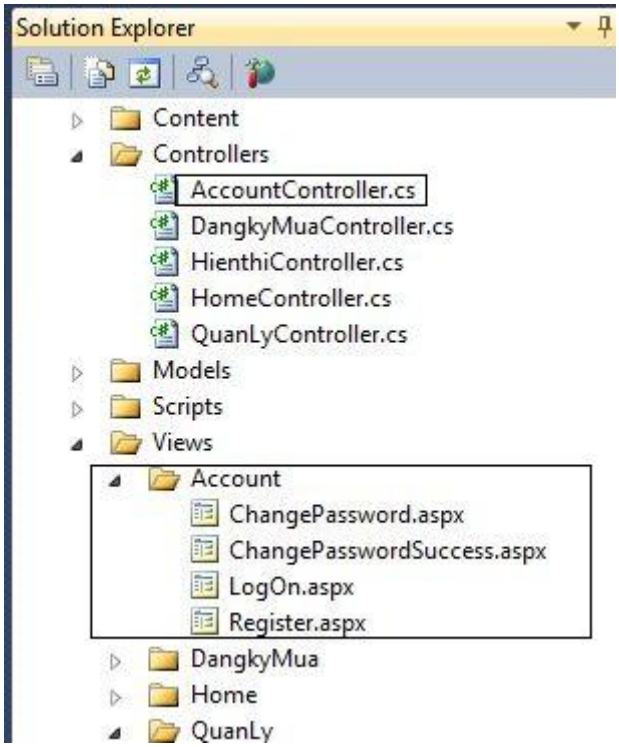
PHẦN 4

BẢO MẬT VỚI ASP.NET MVC APPLICATION

Có nhiều phương pháp xác thực người dùng như Windows Authentication, Forms Authentication. Tuy nhiên trong chương này em chỉ trình bày xác thực dựa trên Forms Authentication, giải thích cách sử dụng Forms Authentication để yêu cầu bằng password cho các View. Sử dụng Website Administration Tool tạo người dùng và phân nhóm người dùng, ngăn chặn những người người trái phép.

- **Tạo người dùng với ASP.NET MVC Application**

Mặc định khi ứng dụng được tạo sẽ có sẵn một Controller có tên AccountController.cs và có các View tương ứng ChangePassword.aspx, ChangePasswordSuccess.aspx, Login.aspx, Register.aspx.



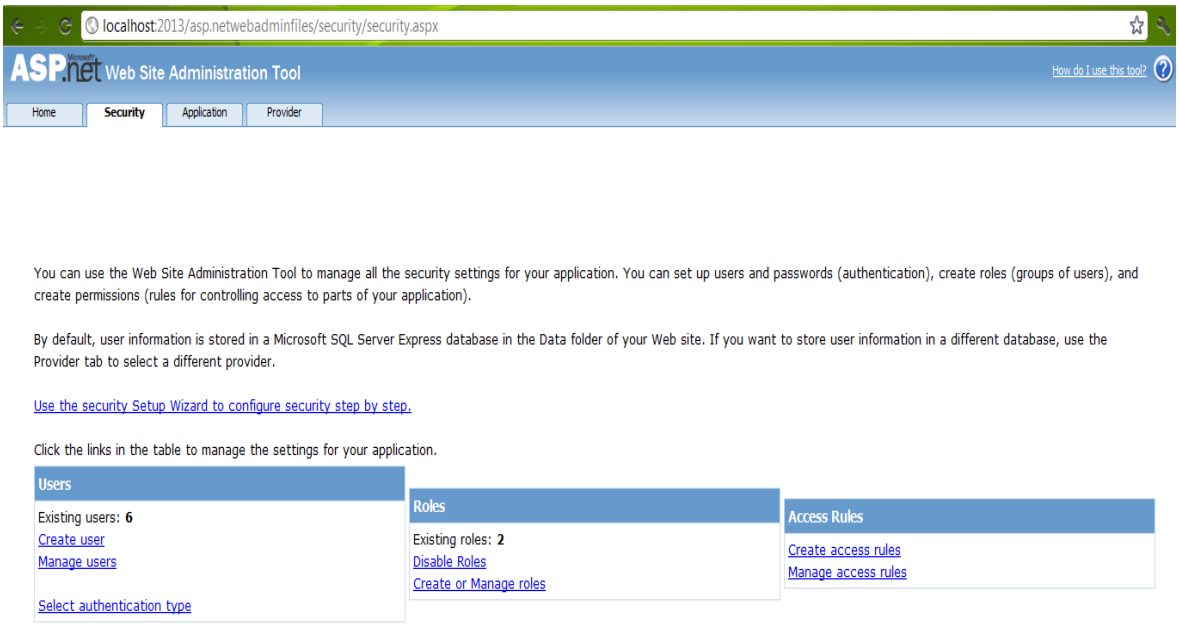
Hình 4.1: Quản lý người dùng

View thể hiện việc đăng ký một người dùng mới như sau:

Hình 4.2: Đăng ký tài khoản mới

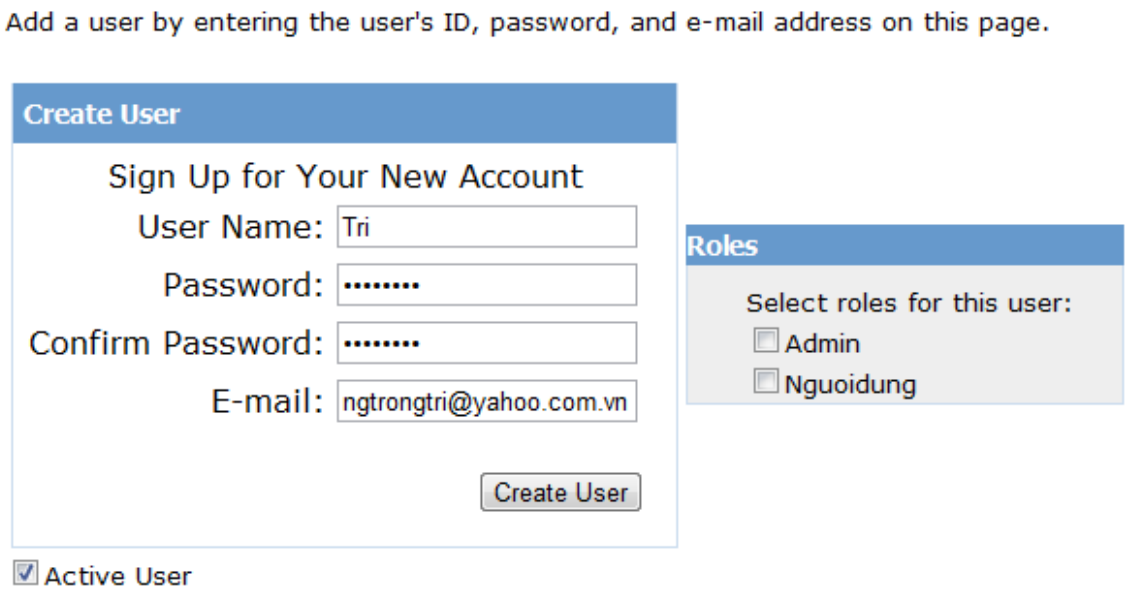
- **Quản lý người dùng với công cụ Website Administration Tool**

Chọn menu Project -> ASP.NET Configuration. Xuất hiện công cụ Website Administration Tool, chọn Tab Security



Hình 4.3: Trang quản trị người dùng

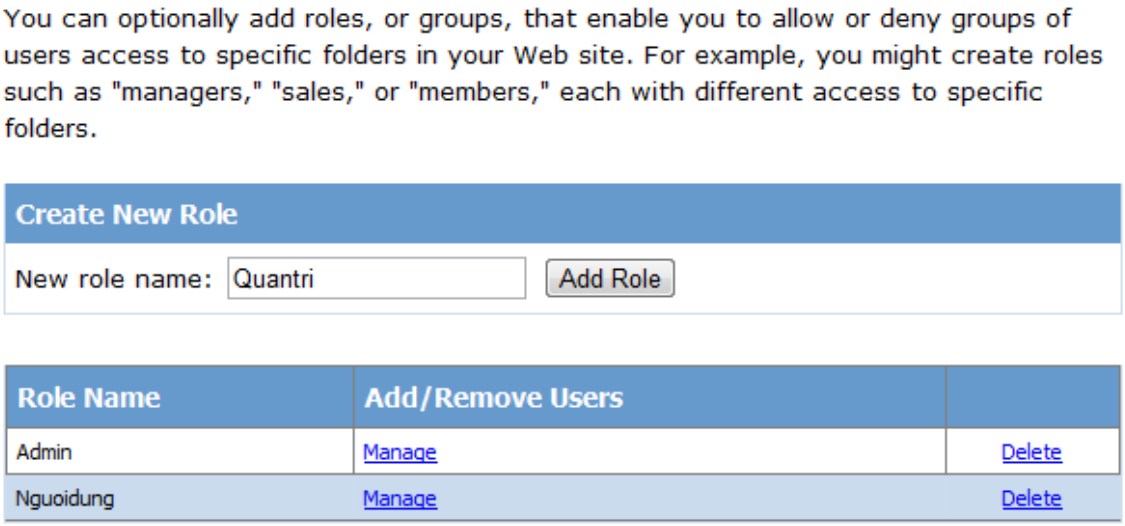
Click và Create User để tạo người dùng, ở đây em ví dụ tạo người dùng tên Tri:



Hình 4.4: Tạo user

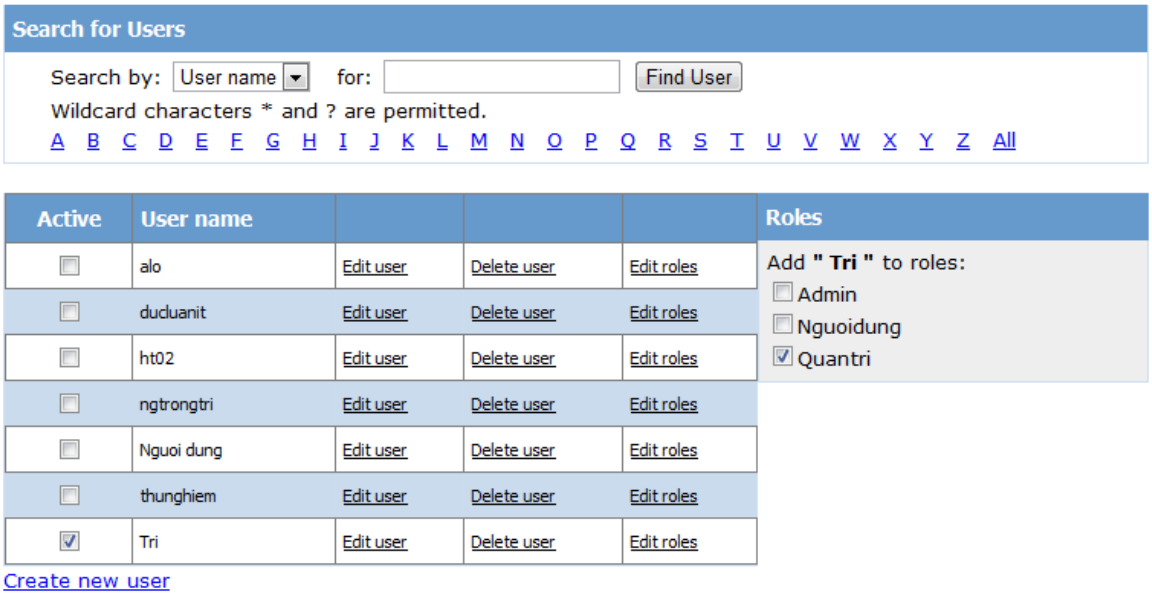
- **Phân quyền nhóm người dùng (Rules)**

Để tạo role trước hết phải enable role bằng cách click vào link Enable roles sau đó click vào Create and Manage roles -> tạo role có tên Quantri.



Hình 4.5: Thêm quyền mới

Bây giờ ta gán quyền cho user Tri mới tạo



Hình 4.6: Gán quyền cho user

Ta áp dụng vào Controller của chúng ta như sau:

```
[Authorize(Roles = "Quantri")]
public ActionResult Deletegt(int id)
{
    Giao_trinh gt= csdl.Giao_trinhs.Where(x => x.Ma_giao_trinh== id).FirstOrDefault();
    if (gt != null)
    {
        csdl.Giao_trinhs.DeleteOnSubmit(gt);
    }

    csdl.SubmitChanges();
    return RedirectToAction("danhsachgt");
}
```

Như vậy khi muốn thực hiện chức năng Delete giáo trình, bắt buộc người dùng phải đăng nhập với quyền "Quantri".

PHẦN 5

CHƯƠNG TRÌNH ỨNG DỤNG

5.1. Mô tả chương trình ứng dụng

- Khảo sát hiện trạng

Trung tâm thông tin tư liệu là một phòng chức năng thuộc trường Việt hàn. Trung tâm có chức năng cung cấp các giáo trình, tư liệu phục vụ việc học tập, nghiên cứu của sinh viên và giảng viên trong trường.

Nhằm minh họa cho phần giới thiệu lý thuyết về ASP.NET MVC, và qua thực tế là hiện nay việc mua giáo trình của trường hiện vẫn còn thủ công. Em đã tìm hiểu và chọn đề tài là xây dựng website đăng ký mua giáo trình trực tuyến dựa trên mô hình ASP.NET MVC nhằm mục đích là giúp cho sinh viên của trường có thể tiến hành việc mua giáo trình một cách dễ dàng và nhanh chóng nhất.

Vào đầu mỗi học kỳ, thư viện thường yêu cầu sinh viên đăng ký mua giáo trình để phục vụ cho việc học tập. Tuy nhiên việc đăng ký chỉ được thực hiện trên giấy tờ, thư viện phát giấy đăng ký cho lớp trưởng và sau đó thu lại. Việc đăng ký thủ công này gây mất thời gian của sinh viên lẫn của thư viện, vì thế việc có một website đăng ký mua giáo trình trực tuyến là một đề xuất phù hợp với tình hình hiện nay của trường.

- Xác lập dự án

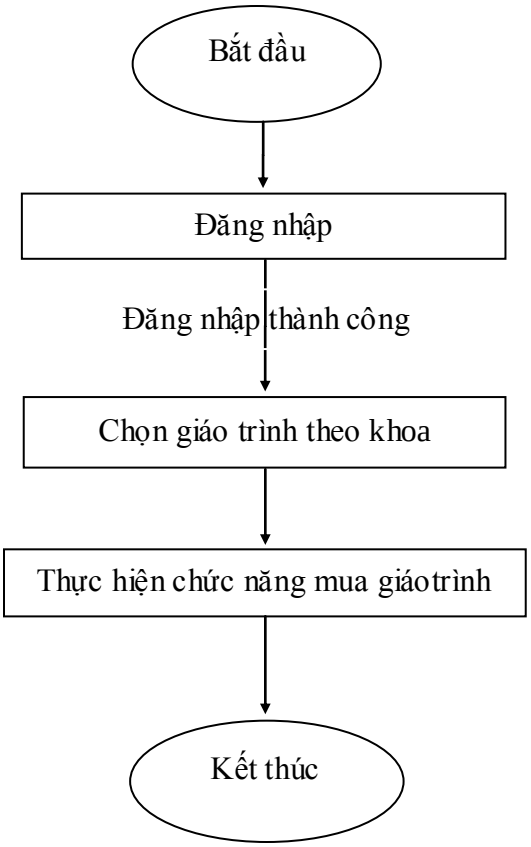
Việc đăng ký mua giáo trình mà một công việc có quy mô nhỏ, nhưng phức tạp. Vì thế, trong chương trình ứng dụng này (được xây dựng với mục tiêu là minh họa cho các lý thuyết được nêu ra trong các chương trên), em sẽ giới hạn lại độ phức tạp của chương trình. Sinh viên sau khi đăng ký thì sẽ thực hiện việc mua giáo trình, và trung tâm sẽ thực hiện việc xác nhận thông tin và cũng cấp giáo trình cho sinh viên.

Quy trình hoạt động của website

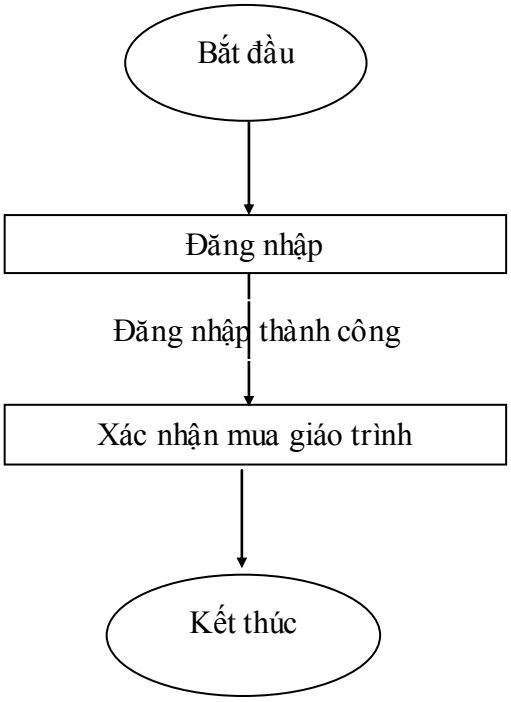
- Về phía trung tâm:
  - Quản lý việc cập nhật giáo trình, khoa.
  - Quản lý việc đăng ký mua của sinh viên.
- Về phía sinh viên:
  - Tiến hành chọn lựa giáo trình dựa trên các khoa tương ứng.
  - Tiến hành điền các thông tin cần thiết và cập nhập thông tin.

➢ Quy trình nghiệp vụ

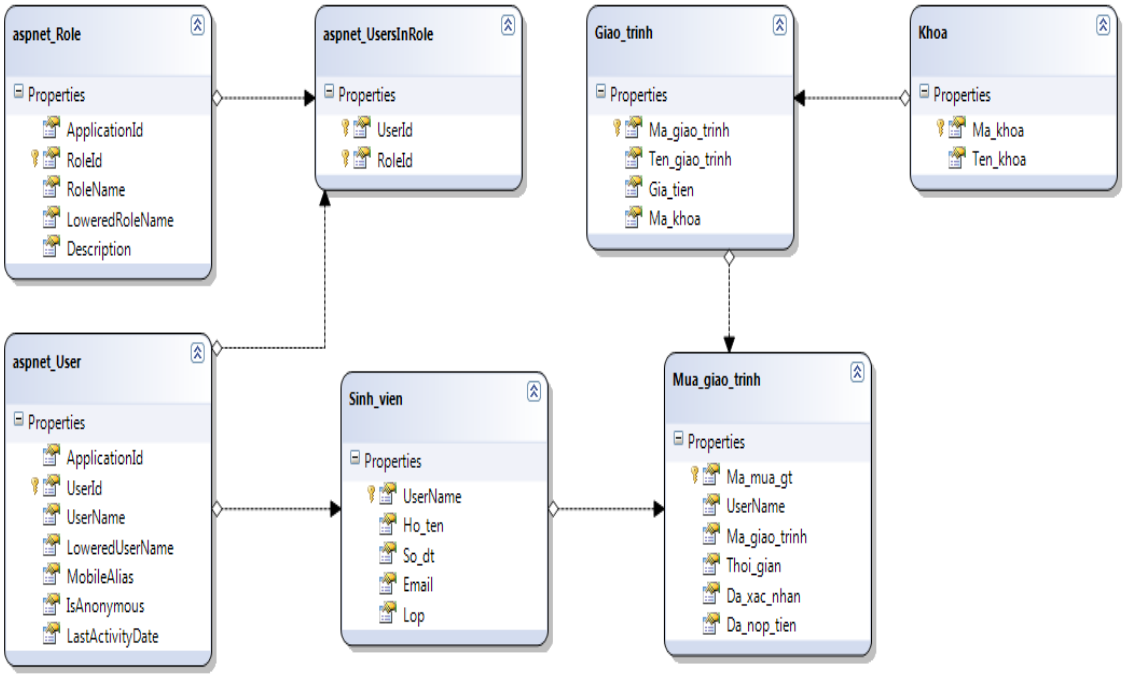
Quy trình đăng ký mua sách



Quy trình xác nhận mua giáo trình



➢ Mô hình cơ sở dữ liệu



Hình 5.1: Mô hình cơ sở dữ liệu

5.2. Hình ảnh các chức năng chính của trang website

➢ Trang chủ: đây là trang chủ của website, sẽ xuất hiện trước tiên khi người dùng truy cập vào.

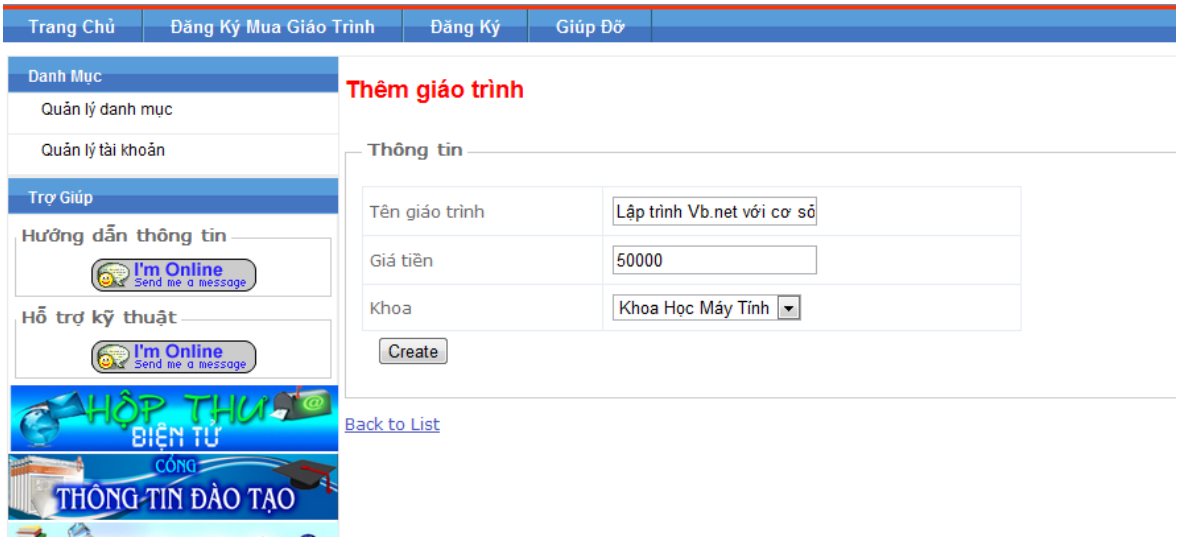
Hình 5.2: Trang chủ website

➢ Trang quản lý khoa: đây là trang dùng để hiện thông tin danh sách các khoa của trường, trung tâm có thể trực tiếp thêm, xóa khoa tại trang này.

Hình 5.3: Trang quản lý các khoa

➢ Trang quản lý giáo trình: Trang dùng để hiển thị danh sách các giáo trình, đồng thời trung tâm có thể xóa giáo trình trực tiếp tại đây. Tại trang này, trung tâm có thể chuyển tới các trang như chỉnh sửa giáo trình, thêm giáo trình.

Hình 5.4: Trang quản lý thông tin giáo trình



Hình 5.5: Trang thêm giáo trình

Hình 5.6: Trang sửa thông tin giáo trình

➢ Trang đăng ký mua giáo trình: Tại trang này, sinh viên có thể lọc giáo trình theo khoa, chọn giáo trình, điền thông tin sinh viên và cuối cùng là hoàn thành việc đăng ký mua giáo trình.

Hình 5.7: Trang đăng ký mua giáo trình

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

➢ Kết luận

Sau gần 3 tháng nghiên cứu và xây dựng, cuối cùng em cũng đã hoàn thành bản báo cáo nghiên cứu lý thuyết và chương trình demo.

Về báo cáo lý thuyết, em đã trình bày những kiến thức cơ bản nhất của ASP.NET MVC, tuy chưa đầy đủ những kiến thức, nhưng em nghĩ rằng những gì em trình bày trong đồ án là những kiến thức nền tảng và quan trọng để mọi người có thể tiếp tục tìm hiểu những kiến thức sâu và mới hơn.

Về chương trình demo, mặc dù quy mô chương trình không lớn, chức năng đơn giản. Nhưng những kiến thức thường dùng nhất của ASP.NET MVC em hầu như đã đưa vào hết, ngoài ra còn có một số kiến thức nâng cao khác. Với demo này, tuy chưa thể áp dụng vào thực tế được, nhưng em tin rằng đây sẽ là một chương trình tham khảo rất tốt cho những bạn muốn học ASP.NET MVC.

➢ Hướng phát triển

Em sẽ bổ sung thêm nhiều chức năng hơn nữa, đồng thời cố gắng để website có thể ứng dụng được vào thực tế tại trung tâm thông tin tư liệu trường Việt Hàn.

TÀI LIỆU THAM KHẢO

[1] Ebook: Wrox - Professional ASP.NET MVC.

[2] Website : http://www.asp.net/mvc.

[3] Website: http://weblogs.asp.